# Context Adaptive Space Quantization for Image Coding

by

Jeffrey Erbrecht

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2016

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

**Abstract**

One of the most widely used lossy image compression formats in the world is JPEG. It operates by splitting an image into blocks, applying a frequency transform to each block, quantizing each transformed block, and entropy coding the resulting quantized values. Its popularity is a results of its simple technical description and its ability to achieve very good compression ratios.

Given the enormous popularity of JPEG, much work has been done over the past two decades on quantizer optimization. Early works focused on optimizing the table of quantizer step sizes in JPEG in an adaptive manner, yielding significant gains in rate-distortion (RD) performance when compared to using the sample quantization table provided in the JPEG standard; this type of quantizer optimization is referred to as hard decision quantization (HDQ). To address the problem of local adaptivity in JPEG, optimization of the quantized values themselves was then considered in addition to optimizing the quantization table; this type of optimization is referred to as soft decision quantization (SDQ). But even SDQ methods cannot fully overcome the problem of local adaptivity in JPEG; nonetheless, the results from SDQ optimization suggest that overcoming this problem has potentially significant gains in RD performance.

In this thesis, we propose a new kind of quantization called context adaptive space quantization (CASQ), where each block in an image is quantized and subsequently entropy coded conditioned on a quantization context. This facilitates the use of different quantizers for different parts of an image. If an image contains regions of varying amounts of detail, for example, then those regions may be assigned different quantization contexts so that they may be quantized differently; then, quantizer optimization may be performed over local regions of an image rather than other the entire image at once. In some sense, CASQ provides the ability to overcome the problem of local adaptivity. We also formulate and solve the problem of quantizer optimization in both the HDQ and SDQ settings using CASQ.

We then propose a practical image coder based on JPEG using CASQ optimization. With our coder, significant gains in RD performance are observed. On average, in the case of Huffman coding under HDQ we see a gain of 1.78 dB PSNR compared to baseline JPEG and 0.23 dB compared to the state-of-the-art method. In the worst cases, our image coder performs no worse than state-of-the-art methods. Furthermore, the additional computational complexity of our image coder when compared to baseline JPEG encoding without optimization is very small, on the order of 150 ms for a $2048 \times 2560$ image in the HDQ case and 4000 ms in the SDQ case.

**Acknowledgements**

I would like to thank my supervisor, Prof. En-hui Yang, for his guidance and support throughout my graduate studies. He has challenged me to think in new ways and to always keep pushing forward.

I would like to thank Prof. Zhou Wang and Prof. Oleg Michailovich for being readers of this thesis, and for the valuable comments and suggestions that they have provided.

I would also like to thank my colleagues in the multimedia communications lab for their continuous support and friendship: Hossam Amer, Jingyun Bian, Nan Hu, Yi Shan, Jiasen Xu, Jin Meng and Xiang Yu. I would like to thank Chang Sun and Longji Wang for graciously providing their SDQ implementations, upon which the experiments in this thesis are based. I would like to thank visiting scholars Xiangwen Wang and Jing He for their interest and encouragement in my research.

Last but not least, I would like to thank my girlfriend Ngan Hoang for her unending support and encouragement, and I would like to thank my mom for supporting me in all of my academic endeavours.

# Table of Contents

# List of Tables

# List of Figures

# List of Abbreviations

| | |
|---|---|
| AC | a DCT frequency position that is not DC (see: DC) |
| ARL | adaptive runlength coding |
| bpp | bits per pixel |
| CASQ | context adaptive space quantization |
| dB | decibels |
| DC | the DCT frequency at position zero; represents the (scaled) average signal level in a block |
| DCT | discrete cosine transform |
| DWT | discrete wavelet transform |
| EOB | end of block |
| HDQ | hard decision quantization |
| HVS | human visual system |
| KKT | Karush-Kuhn-Tucker |
| LPTCM | Laplacian transparent composite modelling |
| MSE | mean-squared error |
| PSNR | peak signal-to-noise ratio |
| RD | rate-distortion |

| | |
|---|---|
| SDQ | soft decision quantization |
| SSIM | structural similarity |

# Chapter 1

# Introduction

## 1.1 Motivation

One of the most widely used lossy image compression formats in the world is JPEG. First published as a standard in 1992 [2], it remains the *de facto* lossy image compression format of choice in all forms of digital entertainment, digital photography, web browsing, and more. Part of its success may be attributed to its ability to compress an image to a fraction of its uncompressed size with little perceivable difference in reconstructed image quality.

JPEG employs a transform coding approach to compression, summarized in Fig. 1.1. First, the input image is partitioned into non-overlapping 8×8 blocks. A discrete cosine transform (DCT) is applied to each of these blocks. Then, the transformed values are quantized into discrete integers in a process called quantization. Finally, the quantized values are further compressed in a lossless process called entropy coding, resulting in a compressed bit stream to be transmitted and ultimately decoded for reconstruction. A JPEG decoder (summarized in Fig. 1.2) simply reverses the entire process to produce a reconstructed image that is a close approximation to the original.

In JPEG, quantization is uniform. The quantizer is represented by a table of 64 step sizes, one for each of the 64 DCT frequency positions. Each DCT coefficient in an 8×8 block is uniformly quantized using the step size corresponding to its frequency position within that block. The quantization process is generally irreversible, meaning that applying the reverse quantization process (called dequantization) will not always yield the original transformed values; this places JPEG into one of two overall classes of coders called *lossy coders*, with the other class being *lossless coders*. Thus, when quantizing the DCT

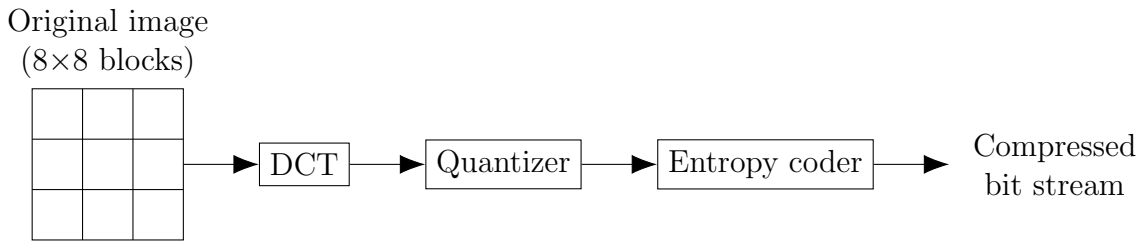



Figure 1.1: Overview of a JPEG encoder

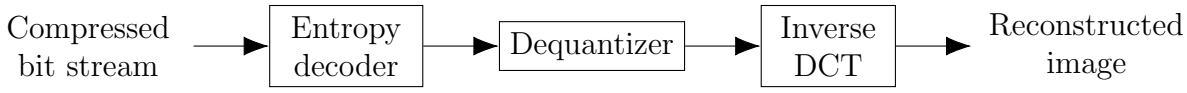



Figure 1.2: Overview of a JPEG decoder

coefficients, some information about the image is lost permanently and the reconstructed image will only be an approximation of the original; in other words, quantization causes the image to incur some kind of distortion. This distortion is typically measured as a quantity called mean-squared error (MSE), which is inversely related to a common image quality measure called peak signal-to-noise ratio (PSNR) in units of decibels (dB); a lower MSE corresponds to a higher PSNR, which corresponds to a higher quality reconstructed image compared to the original.

Not only is quantization the cause of distortion, but it also has the biggest influence on the bit rate of the compressed bit stream, assuming that the entropy coder is efficient. Therefore, quantization is central to the rate-distortion (RD) trade-off in JPEG image compression. Motivated by this centrality, many authors have attempted to solve the problem of finding optimal quantizers (in the RD sense) for JPEG [3–7]. These solutions fall into two categories, namely hard decision quantization (HDQ) and soft decision quantization (SDQ). In HDQ, only the quantization table is optimized. In SDQ, some attempt is also made to optimize the quantized indices themselves, jointly with the quantization table. In [6] specifically, the problem of jointly optimizing both the quantization table and the quantized indices was fully solved, and an efficient algorithm was also given by the authors; this resulted in superior RD performance and remains the state-of-the-art JPEG quantization method.

However, despite even the superiority of the quantization method proposed in [6], a

significant limitation of JPEG remains: only one quantization table may be specified per colour component (equivalently, in the case of single-channel greyscale images, only one table may be specified per image). This is sometimes referred to as the problem of *local adaptivity*: it is not possible to adapt a quantization table to local parts of an image since the same quantization table must be used for every block.

With this limitation in mind, we consider the natural tendency for images to contain local regions of differing perceptual importance. For example, an image might feature a relatively detailed tree with many branches and leaves in the foreground against a rather solid and featureless sky in the background. This foreground region would contain a relatively high amount of energy content in the AC frequencies compared to the background region. When using a quantizer with coarse step sizes, much of this information will be lost resulting in a low quality image; on the other hand, when using a quantizer with fine step sizes, many bits will be spent on the smooth background region with little impact on its perceptual quality. These two competing scenarios are largely irreconcilable as long as JPEG only allows a single quantization table to be defined for all the blocks of an image.

Motivated by this limitation of JPEG, we wish to approach quantization in a fundamentally new way that will not only allow us to quantize different parts of an image in different ways, but also to jointly optimize this quantization over all of the different regions of an image, resulting in potentially significant gains in RD performance.

## 1.2 Contributions

In this thesis, we propose a new kind of quantization dubbed context adaptive space quantization (CASQ), where quantization of each block of an image is conditioned on some kind of context and, furthermore, this quantization is optimized jointly across all step sizes and all contexts. We re-formulate the problem of optimal quantizer design in the HDQ setting and solve it using modifications to the theory presented in [7]. We also re-formulate this problem in the SDQ setting and solve it using modifications to [6] and [8].

To demonstrate our CASQ scheme, we further propose a new image coder based on JPEG, summarized in Fig. 1.3. In addition to CASQ, there are two major blocks in Fig. 1.3 that are designed in order to complete the image coder, which are the *classifier* and the *entropy coder*. We propose an efficient and effective online classifier whose purpose is to assign a quantization context to each block of an image. For simplicity we restrict ourselves to two such contexts, namely *homogeneous* (relatively smooth and featureless) and *non-homogeneous* (relatively detailed); however, the theory of CASQ is presented in a way that

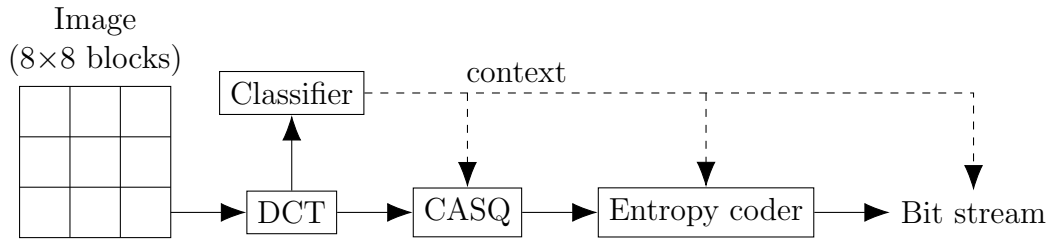



Figure 1.3: Proposed image coding scheme

any number of contexts is permitted. We complete our image coder with two different kinds of entropy coders that are based on JPEG-style Huffman coding and the adaptive runlength coding (ARL) advocated for in [1], respectively.

## 1.3 Thesis Organization

This thesis is organized as follows.

In Chapter 2, we discuss some details of entropy coding in JPEG. We then discuss the problem of optimal quantizer design in the RD sense as a mathematical optimization problem, in both the HDQ and SDQ settings. Finally, we review some previous works on solutions to the optimal quantizer problem in both of these settings.

Then, in Chapter 3, we introduce introduce a new kind of quantization called context adaptive space quantization (CASQ). We introduce the notion of quantization contexts and we reformulate the problem of optimal quantizer design in consideration of these contexts. We then solve this problem in both the HDQ and SDQ settings.

In Chapter 4, we propose a new image coder based on JPEG which demonstrates our CASQ theory using two quantization contexts. We first discuss context selection and propose an efficient and effective online classifier for this purpose. Then, we propose two different entropy coders based on Huffman coding and ARL coding, respectively. Finally, we present experimental results and compare the RD performance and computational performance of the proposed image coder to several benchmarks, including state-of-the-art quantization methods.

Finally, in Chapter 5 we conclude the thesis and discuss some potential applications and future work.

Throughout this thesis, we restrict ourselves to single-channel greyscale images for simplicity. However, the works discussed and theories presented extend trivially to scenarios where multiple colour channels are encoded independently.

# Chapter 2

# Background

In this chapter, we discuss some background topics and works that are relevant to this thesis. We begin with entropy coding in JPEG. We then discuss quantizer optimization in JPEG in both the hard decision quantization (HDQ) and soft-decision quantization (SDQ) settings. Finally, we review some previous works which address quantizer optimization in both of these settings.

## 2.1 Entropy Coding in JPEG

### 2.1.1 Run-Length Coding

The basic operation of a JPEG coder was described in Section 1.1 and summarized in Fig. 1.1. In this subsection, we elaborate a bit more on the run-length coding of quantized DCT coefficients in JPEG.

After the DCT coefficients are quantized, what we are left with is essentially a two-dimensional array of 8×8 blocks of integers. These values must be converted to a one-dimensional sequence for coding. This is done by scanning the blocks in raster order; within each block, the values are scanned in the so-called *zig-zag scanning order* shown in Fig. 2.1. For each block, the DC value is coded separately before the AC values and is not included in the zig-zag scan.

Since most images contain little high-frequency content, there will tend to be a high amount of zeroes in the sequence of quantized values. These zeroes can manifest as long runs of zero values in the zig-zag scanned sequences. To exploit this, JPEG represents

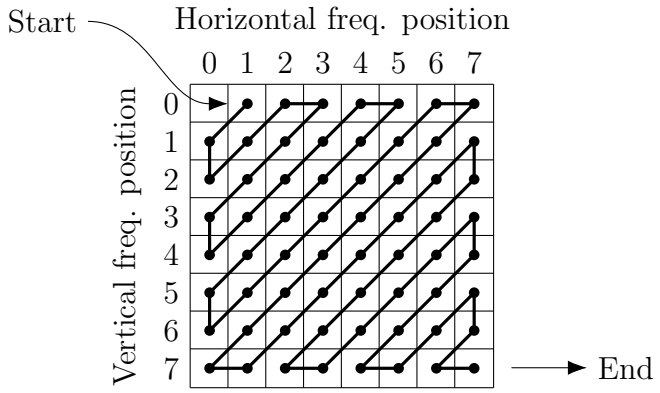



Figure 2.1: Zig-zag scanning order in JPEG

the AC part of the zig-zag sequence from each block as a sequence of run-length codes. Whenever a run of zeroes is encountered in the sequence, it is skipped and represented by a run value indicating the length of the run. This run value (denoted by $r$) is paired with a size value $s_{AC}$ which denotes how many bits are required to represent the first non-zero quantized AC coefficient following the run of zeroes. Immediately following $s_{AC}$ is the quantized index $id$ of the following non-zero AC coefficient, represented as a binary sequence $L_{AC}$ containing the lowest $s_{AC}$ bits of $id$. $r$ and $s_{AC}$ are paired together as a single codeword, denoted by $(r, s_{AC})$ where $r$ and $s_{AC}$ are each represented by four bits, for a total of eight bits. The $AC$ subscript may be dropped from $s_{AC}$ when only the AC coefficients are being considered, which simplifies the run-size pair notation to $(r, s)$.

There are two special run-length codewords, $(15, 0)$ and $(0, 0)$. Since $r$ is represented by four bits, the maximum run length is 15. However, it is possible for a run of greater length to occur since there are 63 AC coefficients in a block. To overcome this problem, the $(15, 0)$ codeword is emitted to indicate that a run of length greater than 15 has occurred, which skips the next 16 zeroes in the run. Several of these codewords may be emitted in succession if the run is very long. In the case where a run of zeroes is encountered that spans the remainder of the block (i.e., there are no remaining non-zero quantized AC coefficients), then a $(0, 0)$ code (called an end-of-block or EOB code) is emitted.

The DC value is treated as the difference between the actual quantized value and the previously coded quantized DC value (from the previously coded block):

$$DIFF := DC(\text{current block}) - DC(\text{previous block}) \tag{2.1}$$

The DC difference $DIFF$ is represented by two values: a size category $s_{DC}$ denoting how

many bits are required to represent $DIFF$, and a binary sequence $L_{DC}$ containing the lower $s_{DC}$ bits of $DIFF$ (or $DIFF - 1$ if it's negative).

### 2.1.2 Huffman Coding

Once all of the run-size sequences $s_{DC}$ and $(r, s_{AC})$ are determined, they are coded into a binary sequence using Huffman coding. There are two Huffman tables defined for this purpose: one for coding the $s_{AC}$ sequence and one for coding the $(r, s_{AC})$ sequence. The DC table is a mapping between each $s_{DC}$ value and a binary codeword. The AC table is a mapping between each eight-bit $(r, s_{AC})$ value and a binary codeword. After each codeword is sent to the bit stream, the corresponding binary $L_{DC}$ or $L_{AC}$ sequence immediately follows.

The Huffman tables are transmitted with the image as side information. Sample Huffman tables are provided in Annex K of the JPEG standard [2]. Better RD performance can be achieved by designing tables that are tailored to the statistics of the actual run-size sequences for the image; an algorithm for designing such tables is given in Section K-2 of the JPEG standard [2].

### 2.1.3 Adaptive Runlength Coding

As an alternative to Huffman coding, a scheme dubbed adaptive runlength coding (ARL) was proposed in [1] by Tu, Liang and Tran. The authors mainly aim to overcome the problem in a JPEG coder of having too many Huffman codewords, resulting in large codeword lengths. Furthermore, the authors aim to exploit certain statistical patterns and correlations commonly found in run-level sequences.

A practical description of the ARL scheme in [1] now follows. In brief, the run and level values are encoded separately using an adaptive binary arithmetic coder with contexts. Three different kinds of values can be encoded: DC values, $RUN$ values and $LEVEL$ values. Each of these values is associated with set of contexts from which a context is chosen for arithmetic coding.

The binarization of each value to be sent to the arithmetic coder is exemplified in Table 2.1. To binarize a $RUN$ value, $RUN + 1$ "0"s are sent to the arithmetic coder followed by a "1". In the special case of an EOB code, a single "1" is sent to the coder. To binarize a $LEVEL$ value, a sign bit is first sent to the coder, followed by $|LEVEL| - 1$ "0"s and a single "1", where $|\cdot|$ denotes the magnitude.

Table 2.1: Binarization of $RUN$ and $LEVEL$ values [1]

| $RUN$ | Binarization | $LEVEL$ | Binarization |
|---|---|---|---|
| EOB | 1 | 1 | 0 1 |
| 0 | 0 1 | -1 | 1 1 |
| 1 | 0 0 1 | 2 | 0 0 1 |
| 2 | 0 0 0 1 | -2 | 1 0 1 |
| 3 | 0 0 0 0 1 | 3 | 0 0 0 1 |
| 4 | 0 0 0 0 0 1 | -3 | 1 0 0 1 |
| ⋮ | ⋮ | ⋮ | ⋮ |

The DC value is predicted as the mean of the reconstructed DC values of the upper and left neighbouring blocks; it is actually the residue from this prediction that is encoded. First, one bit is encoded indicating whether or not this residue is zero. For this bit, one of three context models is used. Define $z$ as the total number of neighbouring blocks (upper and left) that have non-zero quantized residues. Then, the three context models are denoted by

$$(z = 0)(z = 1)(z = 2) \tag{2.2}$$

If the residue is indeed non-zero, then it is coded in the same manner as a $LEVEL$ value.

Next, define $f$ as the number of neighbouring blocks (upper and left) that contain non-zero quantized AC coefficients. To code the first bit of the first $RUN$ value in a block, one of the following three contexts is used:

$$(f = 0)(f = 1)(f = 2) \tag{2.3}$$

Two further models are used for coding the second bit and the remaining bits, respectively, of the first $RUN$ value.

Subsequent $RUN$ values use contexts based on the magnitude $m$ and AC index (in zig-zag scanning order) $l$ of the previously coded $LEVEL$ value. Five contexts are defined for this purpose, for coding the first bit of the current $RUN$ value.

$$\begin{aligned} &(l < 6 \text{ and } m = 1)(l < 6 \text{ and } m > 1) \\ &(6 \leq l < 15 \text{ and } m = 1)(6 \leq l < 15 \text{ and } m > 1) \\ &(l \geq 15) \end{aligned} \tag{2.4}$$

Two more copies of these five models are defined for the second bit and remaining bits, respectively, of the $RUN$ value; thus, the total number of context models for encoding a $RUN$ value is twenty.

To code a $LEVEL$ value, one model is used when encoding its sign bit. For the next bit, four contexts are defined based on the AC index $l$ of the current $LEVEL$ value and the value $r$ of the corresponding $RUN$:

$$\begin{aligned}&(l = 0)(0 < l < 3)(3 \leq l < 15 \text{ and } r < 3)\\&(15 \leq \text{ or } r \geq 3)\end{aligned} \tag{2.5}$$

The remaining bits use another set of the four models described above. Thus, a total of nine contexts are used to code a $LEVEL$ value. In total, 32 contexts are used in ARL coding.

The authors of [1] report a gain in PSNR performance of 1.8 dB on average when compared to baseline JPEG Huffman coding with comparable complexity.

## 2.2 Quantizer Design in JPEG

The JPEG standard permits a customized quantization table to be transmitted as side information along with the compressed bit stream. The standard also provides a sample quantization table that can be used for any image; however, as this quantization table is static and independent of the image being coded, it tends to perform poorly in the RD sense. Thus, we have both the ability and the desire to design a customized quantization table that performs optimally in the RD sense.

In the following discussion, it is important to note the difference between a *quantization table* and a *quantizer*. A *quantization table* is merely a collection of step sizes that the decoder must use to reconstruct the DCT coefficients from a sequence of quantized indices. A *quantizer* is essentially a process that takes a sequence of DCT coefficients as input and produces a sequence of quantized indices as output, along with an appropriate quantization table for the purpose of reconstruction; the means by which these quantized indices are determined are not necessarily bound to any particular mapping.

The problem of designing an optimal quantizer can be approached in two different settings: hard decision quantization (HDQ) and soft decision quantization (SDQ). These two approaches are discussed in the following subsections.

### 2.2.1 Hard Decision Quantization

Hard decision quantization (HDQ) describes a quantization setting where the quantized index for a given DCT coefficient is the output of a simple mapping based only on the coefficient value and the step size corresponding to its frequency position. This mapping can be described by the following equation, where $C_i$ denotes the DCT coefficient at the $i$th frequency position (in zig-zag order) of the current block, $q_i$ denotes the step size for the $i$th frequency position, and $K_i$ denotes the resulting quantized index:

$$K_i = \text{round}\left(\frac{C_i}{q_i}\right) \tag{2.6}$$

Since the DCT coefficients are fixed, then the free parameter to be optimized is the quantization table $Q = \{q_i\}$, $0 \leq i < 64$. The problem of finding the optimal $Q$ in the RD sense is typically formulated as the following constrained optimization:

$$\inf_{Q} R(Q) \quad \text{s.t.} \quad D(Q) \leq D_T \tag{2.7}$$

In (2.7), $R(Q)$ denotes the bit rate resulting from using $Q$ to quantize and subsequently compress the image, $D(Q)$ denotes the corresponding distortion incurred from quantization using $Q$, and $D_T$ is a distortion budget. Equation (2.7) can equivalently be written as

$$\inf_{Q} D(Q) \quad \text{s.t.} \quad R(Q) \leq R_T \tag{2.8}$$

where $R_T$ denotes a bit rate budget. The problem may also be converted to an unconstrained optimization using a Lagrange multiplier $\theta$, where $\theta$ represents the rate-distortion trade-off and is equivalent to a rate or distortion budget:

$$\inf_{Q} \left(R(Q) + \theta D(Q)\right) \tag{2.9}$$

Since the only parameter being optimized is $Q$, then a solution to (2.7) or (2.8) will be a quantization table with step sizes chosen optimally in the RD sense. Previous works which propose solutions to this problem will be discussed in Section 2.3.

### 2.2.2 Soft Decision Quantization

In soft decision quantization (SDQ), the quantized indices themselves are also treated as a free parameter, in addition to $Q$. This is facilitated by a mapping $Q'_Q$ which maps a

sequence of transform coefficients of length $n$ at the $i$th frequency position to a sequence of quantized indices of length $n$ to be reconstructed using the quantization table $Q$:

$$Q'_Q : \mathbb{R}^n \longrightarrow \{0, \pm 1, \pm 2, ...\}^n \tag{2.10}$$

Given an entropy coding method $\phi$, the problem of finding an optimal quantizer may then be formulated as

$$\inf_{Q} \inf_{Q'_Q} R_\phi \left(Q, Q'_Q\right) \quad \text{s.t.} \quad D\left(Q, Q'_Q\right) \leq D_T \tag{2.11}$$

where $R_\phi \left(Q, Q'_Q\right)$ denotes the bit rate resulting from using $\phi$ to code an image quantized by $Q$ and $Q'_Q$, $D\left(Q, Q'_Q\right)$ denotes the corresponding distortion from quantization, and $D_T$ is a distortion budget. This can also be converted to an unconstrained optimization using a Lagrange multiplier $\theta$, much like in the HDQ case:

$$\inf_{Q} \inf_{Q'_Q} \left[R_\phi(Q, Q'_Q) + \theta D(Q, Q'_Q)\right] \tag{2.12}$$

In the specific case of JPEG coding, one can change $Q'_Q$ to the quantized run-size-index sequence $(r, s, id)$ and, furthermore, one can also include the Huffman table $H$ as a free parameter rather than fixing it as $\phi$. This problem can be formulated as follows:

$$\inf_{(r,s,id),H,Q} R((r, s), H) \quad \text{s.t.} \quad D((r, s, id)_Q) \leq D_T \tag{2.13}$$

where $R((r, s), H)$ denotes the bit rate resulting from compressing $(r, s, id)$ using $H$, and $D((r, s, id)_Q)$ denotes the distortion between the original image and the image reconstructed from $(r, s, id)$ using $Q$. Note that $id$ is omitted from the rate calculation because it is not necessary: the bit rate can be entirely determined from $(r, s)$ since the number of bits transmitted for $id$ can be determined by $s$. The quantization table $Q$ is also omitted from the rate calculation because the quantized image is already captured in $(r, s, id)$. Just as before, (2.13) can be converted to an unconstrained optimization using a Lagrange multiplier $\theta$:

$$\inf_{(r,s,id),H,Q} \left[R((r, s), H) + \theta D((r, s, id)_Q)\right] \tag{2.14}$$

Since more free parameters (in addition to $Q$) are being optimized in (2.11), then an optimal solution to (2.11) will generally perform better in the RD sense compared to an optimal solution to (2.7). However, the solution to (2.11) is often more complex by comparison. Therefore, it is worthwhile to investigate solutions to both the HDQ and the SDQ quantization problems: HDQ for its relative simplicity and SDQ for its superior RD performance. Previous work relating to SDQ is also discussed in Section 2.3.

## 2.3 Previous Work on Quantizer Optimization

Many authors have attempted to solve the problem of quantization table optimization in both the HDQ and SDQ settings. In this section, we review some of the major works in these two areas.

### 2.3.1 HDQ Methods

#### Locally Optimized Search using Laplacian Modelling

In [3], Hung and Meng proposed a gradient search method for finding locally optimal quantizer step sizes. The authors used Laplacian distributions to model the statistics of the DCT coefficients; thus, the expressions for approximate bit rate and distortion can easily be calculated.

The algorithm begins by choosing an initial $Q$ with small step sizes so that the actual distortion (mean-squared error, or MSE) from using $Q$ is much smaller than some target distortion. Next, the algorithm calculates the slopes $\partial D/\partial q_i$ and $\partial R/\partial q_i$ for each frequency position $i$, where $D$ and $R$ are computed at both $q_i$ and $q_i + 2$. The $i$ which maximizes the ratio $\partial D/\partial R$ is then found by a simple search, and the corresponding step size in $Q$ is then updated. As a result, the total distortion is increased minimally for some decrease in bit rate. This process is iterated until the target overall distortion is met.

The authors of [3] reported a gain of around 0.5 dB – 1 dB in PSNR performance when compared to using the default quantization table given in the JPEG standard. However, the performance depends highly on the accuracy of the Laplacian model which, while a reasonable choice of distribution, does not exactly fit the statistics of the image.

#### Locally Optimized Search using Real Coding Rates

To overcome the problem of modelling accuracy, an algorithm was given in [4] by Wu and Gersho that is based on the actual statistics of the image. The overall idea of the algorithm is similar to that in [3]: update one step size at a time which optimizes some RD trade-off until a target is met. There are three main differences in the algorithm presented in [4]. First, a rather trivial difference: the algorithm targets a rate budget rather than a distortion budget, so the initial quantization table actually contains large step sizes which are decreased at each iteration. Second, the optimal step size update is actually found at each iteration, rather than using only a fixed difference of 2 as in [3].

Third, and most importantly, the actual coding rate $R$ is used instead of a statistics-based approximation. Consequently, the bit encoding process must be simulated at each iteration in order to compute the real coding rate, resulting in a significant increase in computational complexity.

The authors reported an RD performance which slightly exceeds that of [3]; however, since the actual bit rate is computed at each iteration, the algorithm performs very poorly in the computational sense.

### RD-OPT

In [5], Ratnakar and Livny described their so-called *RD-OPT* algorithm. The authors chose to estimate the rate using the empirical entropy of the quantized coefficients, rather than any specific statistical model or full coding simulation. The distortion is closely approximated using a histogram-based approach, where the DCT coefficients of each block are grouped into specially-chosen bins. The bins are chosen such that every value grouped into a particular bin is always quantized to the same value; then, the DCT coefficient values can be approximated by the midpoint value of the bin in which they are placed, and thus the distortion can easily be computed without scanning the entire image at each iteration (aside from an initial one-time grouping of coefficients into bins). The authors showed that the root-mean-square error of this approximate distortion is within $\pm 0.25$ of the actual root-mean-square error. The entropy calculation for the rate is done in a similar way.

The authors then proposed an efficient binary search algorithm to minimize the Lagrangian in (2.9). Furthermore, if an actual rate budget is desired, then the $\theta$ corresponding to that budget can efficiently be found from a bisection search. As a result of this efficient approach, the authors reported RD performance similar to [4] with much lower computational complexity.

### OptD

The RD-OPT algorithm described in [5] remained the state-of-the-art HDQ method until the so-called *OptD* theory was proposed in [7] by Yang, Sun and Meng. OptD theory is actually formulated in the setting of SDQ. Specifically, it attempts to solve for $Q$ in (2.11) without explicitly solving for $Q'_Q$ by jointly optimizing all of the step sizes in $Q$. However, since only $Q$ is actually being found by OptD, then in some sense it may still be viewed as a solution to quantization table design under HDQ, despite being formulated under SDQ.

The authors of [7] also presented an efficient algorithm to compute a very close approximation to the optimal $Q$ based on Laplacian modelling of the DCT coefficients. Interestingly, the $Q$ generated by this algorithm is superior to that of RD-OPT in the RD sense with only a fraction of the computational complexity: the authors reported an average gain of 0.5 dB PSNR performance with complexity reduced by a factor of more than 2000 when compared to RD-OPT. Furthermore, OptD theory is presented in a manner that makes it easily compatible with any efficient block-based coding method with any number of step sizes in $Q$. With all of these things considered, we choose OptD theory as the basis for our CASQ method in the HDQ setting to be formulated in Chapter 3.

### 2.3.2 SDQ Methods

#### Thresholding

A sub-optimal SDQ technique called *thresholding* or *zeroing* was investigated in [9] by Ramchandran and Vetterli. The goal of thresholding is to drop (or threshold) coefficients to zero in order to save bits with minimal impact on the resulting distortion.

The authors noted that the optimal solution to the RD optimization can be found independently on a per-block basis; thus, their analysis was limited to one block, which can be repeated for each block in the image to find an overall solution. The optimization problem was presented as follows:

$$\min_{S \subseteq T} D(S) \quad \text{s.t.} \quad R(S) \leq R_T \tag{2.15}$$

where $T = \{0, 1, ..., 63\}$ is the set of all possible coefficient indices in a block in zig-zag scanning order, $S$ is a subset of $T$, $D(S)$ is the distortion resulting from retaining the coefficients indexed by $S$ and $R(S)$ is the corresponding rate, and $R_T$ is a rate budget. This is converted to an unconstrained optimization via the Lagrange multiplier $\lambda$:

$$\min_{S} \left[ J(\lambda) = D(S) + \lambda R(S) \right] \tag{2.16}$$

The authors then proposed an efficient dynamic programming algorithm to find $S$ for each block. If a rate or distortion budget is required, then the corresponding $\lambda$ can be efficiently found by a bisection search.

Note that this algorithm only finds the optimal set of coefficients $S$ to threshold to zero. The quantization table $Q$ is actually fixed as the example table from the JPEG standard, scaled by some constant. Thus, this method does not optimize the quantization table step sizes at all.

### Joint Thresholding and Table Optimization

The problem of jointly optimizing both the set of threshold coefficients and the quantization table was addressed in [10] by Crouse and Ramchandran. The authors formulated the problem as the following minimization:

$$\min_{T,Q,H} D(T,Q) \quad \text{s.t.} \quad R(T,Q,H) \leq R_T \tag{2.17}$$

In the problem above, $T$ now refers to a binary mapping that signals whether each DCT coefficient should be quantized as normal (represented by a "1") or instead be thresholded to zero (represented by a "0"); not to be confused with the definition of $T$ from the previous discussion on thresholding. The remaining parameters $Q$ (quantization table), $H$ (Huffman table), $D$ (distortion), $R$ (bit rate) and $R_T$ (bit rate budget) are all consistent with their previous definitions.

The problem in (2.17) is converted to an unconstrained problem via the Lagrange multiplier $\lambda$:

$$\min_{T,Q,H} \left[ J(\lambda) = D(T,Q) + \lambda R(T,Q,H) \right] \tag{2.18}$$

where $J(\lambda)$ denotes the Lagrangian cost associated with the RD trade-off parameter $\lambda$.

A sub-optimal, iterative solution that converges to a local minimum was proposed. This solution works by first fixing an initial quantization table $Q$, threshold table $T$, and Huffman table $H$. Then, each of these three parameters is updated sequentially to a more optimal choice by fixing the other two parameters; this process repeats iteratively until some convergence criterion is met. To update $Q$ given that $T$ and $H$ are fixed, the authors use a method similar to that in [4]. To update $T$, the same method from [9] is used. $H$ is updated based on the statistics of the quantized image given that $Q$ and $T$ are fixed. The algorithm only converges to a local optimum due to the initial $Q$, $T$ and $H$ being fixed to somewhat arbitrary values. Despite this sub-optimality, the authors reported an average gain in PSNR performance of about 1 dB when compared to [9]. The authors of [10] briefly discussed the possibility of allowing DCT coefficients to be quantized to *any* level in order to achieve maximal gain in the RD sense, rather than simply thresholding some coefficients to zero; however, they also remind us that when the quantization table $Q$ is fixed, then simply thresholding coefficients to zero performs approximately as well as allowing them to be quantized to arbitrary levels.

### Full SDQ Solution

The possibility of allowing quantized values to be changed to arbitrary levels for RD performance gain in the JPEG setting was finally explored in full in [6] by Yang and Wang. The authors of [6] proposed a solution to (2.13). Similarly to [10], the authors of [6] proposed an efficient iterative solution to (2.13) that converges to a local optimum. The main differences are that the optimization of $(r, s, id)$ involves a *full* search, thus allowing quantized values to be changed to any level, rather than only possibly dropping them to zero; also, the iterative step which updates $Q$ is much more efficient compared to the method used in [10]. The solution in [6] works as follows. First, since the probability distribution $P$ of run-size pairs $(r, s)$ completely determines the Huffman table $H$, then $H$ may be replaced by $P$ in (2.13). The empirical entropy of this distribution is then used to estimate the bit cost of each run-size pair. This change is justified if custom Huffman tables are used, because then actual bit cost is very close to the empirical entropy. Next, an initial $Q$ is chosen and the image is subsequently quantized using the initial $Q$ in order to determine an initial run-size distribution $P$. The core part of the algorithm now begins: fix $Q$ and $P$ and optimize $(r, s, id)$ (referred to as "Step 2" by the authors of [6]), which is written as the following minimization:

$$\min_{(r,s,id)} \left[ J(\lambda) = D((r, s, id)_Q) + \lambda R((r, s), P) \right] \tag{2.19}$$

Then, fix $(r, s, id)$ and optimize $Q$ and $P$ (referred to as "Step 3"), which is written as:

$$\min_{(Q,P)} \left[ J(\lambda) = D((r, s, id)_Q) + \lambda R((r, s), P) \right] \tag{2.20}$$

These two steps iterate back and forth until a convergence criterion is met. At the end, a customized Huffman table is generated based on the final run-size statistics of the quantized image.

The authors of [6] proposed an efficient graph-based dynamic programming algorithm to solve the minimization in Step 2. The algorithm operates on each block independently, allowing for an efficient parallel implementation. For each block, a graph is created with 65 nodes: 64 corresponding to the DCT indices of a block, and an additional block denoting an end-of-block (EOB) code. A set of paths is then generated between every possible pair of nodes that includes every possible run-size transition between those two nodes. Each path from node $i$ to node $j$ corresponding to a size group $s$ has an associated incremental Lagrangian cost $J$ for quantizing the DCT value at node $j$ into size group $s$ and quantizing all nodes in between nodes $i$ and $j$ to zero. This cost is computed using the MSE distortion metric and the empirical entropy of run-size pairs for rate estimation. A dynamic

programming algorithm then searches for the path from the first node to the end node which minimizes the sum of incremental Lagrangian costs. This path is equivalent to a full run-size-index sequence for the corresponding block, which in fact represents the optimal quantized version of that block for a fixed $Q$.

Step 3 is easily solved if $D(\cdot)$ is the squared-error distortion metric. Recall that Step 3 optimizes $Q$ and $P$ only, given that $(r, s, id)$ is now fixed from Step 2. Note that the rate does not depend on $Q$ once $(r, s)$ is fixed. Therefore, only the distortion term actually remains in the minimization for Step 3 which, being convex, continuous and differentiable, has a trivial closed-form solution for $Q$. The run-size probability distribution $P$ is directly computed from the fixed $(r, s)$.

The full SDQ solution from [6] is the current state-of-the-art JPEG quantization method with full baseline compatibility, in that it has the best RD performance out of all the HDQ and SDQ methods investigated so far. It performs even better than wavelet-based schemes in many cases. The only sub-optimality of the full SDQ solution comes from the choice of the initial quantization table, which was addressed in [7]. The problem of finding a globally optimal $Q$ in the SDQ setting remains an open problem.

In addition, the authors of [6] also solved a similar problem for SDQ in a JPEG-style setting where adaptive runlength coding (ARL) is used instead of Huffman coding [8]. In ARL coding (to be described in detail in Section 2.1.3), $RUN$ and $LEVEL$ values are coded separately ($r$ and $l$ for short) using a context-based binary arithmetic coder. In [8], the problem of jointly optimizing $Q$ and $(r, l)$ over all possible values and all possible context models $M$ is formulated using the Lagrange multiplier $\lambda$ as

$$\min_{(r,l),M,Q} [J(\lambda) = D((r, l)_Q) + \lambda R((r, l), M)] \tag{2.21}$$

The process of solving (2.21) is largely based on the exact same ideas from the Huffman case, so the details are omitted here. The reader is referred to [8] for a complete description.

## 2.4 Summary

In this chapter, we discussed the run-length coding and Huffman coding methods of baseline JPEG, as well as an alternative method to Huffman coding called ARL coding. We then introduced the problem of optimal quantizer design in both the HDQ and SDQ settings. Finally, we reviewed some past works, including current state-of-the-art methods, on solving this problem under HDQ and SDQ. All of these topics lead into the theories presented in the next chapter.

# Chapter 3

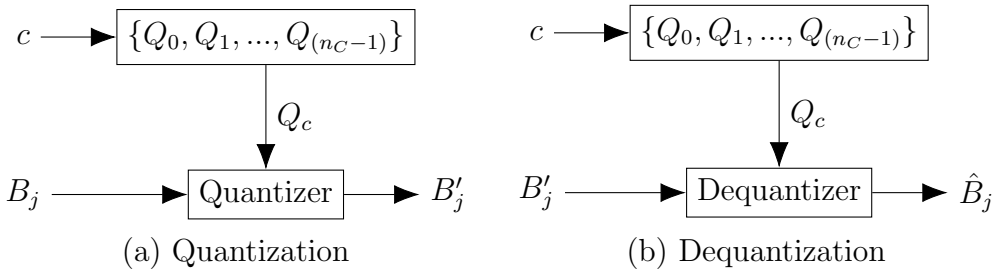
# Context Adaptive Space Quantization

## 3.1 Overview

In Chapter 1, we briefly discussed a major limitation of JPEG: that it lacks local adaptivity. Even though custom quantization tables and Huffman tables may be defined and optimized (for example, using the methods discussed in Section 2.3.1), they must be applied in the same manner to every single block in the image. This is one of the major motivations for SDQ methods, where local adaptivity can be achieved to some extent by optimizing the quantized DCT coefficients themselves [6, 8–10].

But even the SDQ methods discussed in Section 2.3.2 are still limited by the lack of local adaptivity supported by JPEG. Motivated by this, we propose a new approach to image quantization called context adaptive space quantization (CASQ). In CASQ, quantization (and dequantization) of a given block of DCT coefficients is performed conditioned on a *quantization context*. More specifically, a quantization context is a piece of side information that is associated with each block prior to quantization, so that quantization itself may be *conditioned* on that context, allowing different blocks to be quantized (and subsequently dequantized) in different ways.

A novel way to demonstrate this concept for a JPEG-style image coder in the HDQ setting is to define an arbitrary number $n_C$ of quantization contexts and associate each of them with its own quantization table $Q_c$, $0 \leq c < n_C$. Suppose that there are $n_B$ blocks in an image, indexed from 0 to $n_B - 1$. A context index $c$ is then selected for each block $B_j$,




(a) Quantization (b) Dequantization

Figure 3.1: General CASQ process

$0 \leq j < n_B$, in any manner desired. Quantization of $B_j$ into $B'_j$ would then use $Q_c$. The same process is followed to dequantize $B'_j$ into $\hat{B}_j$ using $Q_c$, where $\hat{B}_j$ is the reconstructed approximation of $B_j$ at the decoder side. This is summarized in Figs. 3.1(a) and 3.1(b).

In the extreme case, one could define a separate context for each block; that is, $n_C$ is equal to the number of blocks in the image, and the $j$th coded block would use the $j$th context for quantization/dequantization. Then, we would have direct control over the quantizer step size for every single coefficient in the image, potentially allowing for superior RD performance. Obviously, there would be an enormous amount of overhead in having to transmit the definition of each quantization table to the decoder; nonetheless, this example demonstrates one particular way for CASQ to operate.

A more realistic scenario would be to define a small number of contexts and then group regions of blocks into those contexts according to some rule. For example, one could define a "flat, smooth" context for all blocks that are relatively featureless, and then associate with that context a quantization table with coarse step sizes. One could then group all remaining blocks into a second "detailed" context, which could be associated with a quantizer with fine step sizes. If the contexts are selected properly and the quantizers are all optimized jointly, then some gain may be had in the RD sense compared to having only a single quantizer applied to all blocks unconditionally as in JPEG. The gain would come from being able to spend fewer bits on quantizing the relatively featureless region of blocks, while simultaneously preserving more information in the detailed region.

This two-context approach will be explored in a practical image coding system later in Chapter 4, which includes an efficient and effective classifier for the purpose of context selection. For the remainder of this chapter, we leave the exact definitions of the contexts as an implementation detail; we will simply assume that there are $n_C$ contexts in total, indexed from 0 to $n_C - 1$. We will assume that context selection for each block is done prior to quantization/dequantization, is fixed, and is known to the encoder and decoder. The

problem of optimizing context selection itself is discussed as a future work in Section 5.2.2.

Based on the discussion in Chapter 2, an important question arises: under CASQ in the HDQ setting, how do we formulate the problem of optimal quantization table design (in the RD sense) across all contexts and all step sizes jointly? In the SDQ setting, how do we formulate this problem across all contexts, quantization tables, quantized coefficients and entropy coders jointly? In the following sections, we will formulate these problems and subsequently solve them with efficient and computationally cheap solutions based on some of the previous works described in Chapter 2.

## 3.2 CASQ Design: Problem Formulation

In this section, we formulate the problem of optimal quantizer design (in the RD sense) under CASQ across all contexts jointly.

### Definitions

Assume that we have an input image $I_0$ that is divided into evenly spaced non-overlapping blocks, where the size of each block is $N \times M$. Let $n_B$ denote the total number of blocks. Let $L = NM$ denote the number of coefficients contained within each block (for example, $L = 8 \cdot 8 = 64$ in the 8×8 DCT setting of JPEG). Assume that some $N \times M$ transform has been applied to each block in $I_0$ (for example, the 8×8 DCT). Note that $L$ is also equal to the number of transform frequency positions within a block.

Define $n_C$ contexts indexed from 0 to $n_C - 1$. For the $c$th context, we define a corresponding quantization table $Q_c$ for the purpose of uniform reconstruction; hence, $Q_c$ may be represented by its step sizes via

$$Q_c = \{q_{c0}, q_{c1}, ..., q_{c(L-1)}\} \tag{3.1}$$

Assume that each $Q_c$ contains the same number of step sizes $L$. Uniform reconstruction in this scenario means that given a quantization context index $c$ and a quantized index $u \in \{0, \pm 1, \pm 2, \pm 3, ...\}$ at the $i$th frequency position, the corresponding reconstructed value will be $u \cdot q_{ci}$. Note that the frequency position in $Q_c$ is indexed one-dimensionally, while in general each block is a two-dimensional array of coefficients. It is assumed that some one-to-one mapping exists between the two-dimensional frequency positions in a block and the one-dimensional frequency positions in $Q_c$ (for example, by the zig-zag scanning order

in JPEG). We will use one-dimensional frequency indexing throughout this chapter for simplicity.

We define an overall quantization table $Q$ as the collection of per-context quantization tables:

$$Q = \{Q_0, Q_1, ..., Q_{n_C-1}\} \tag{3.2}$$

Note that because each block is associated with one of the $n_C$ contexts, then each frequency position within a block is also associated with that context. Thus, there are a total of $n_C \cdot L$ possible combinations of transform frequency position and context. Accordingly, the number of step sizes contained in $Q$ is also $n_C \cdot L$; we can then also view $Q$ as a single quantization table containing $n_C \cdot L$ step sizes.

Whenever the term "context" is used after this point, we are referring to *quantization context* unless otherwise specified. A distinction only needs to be made whenever arithmetic coding is being discussed alongside quantization, in which case we will refer to *quantization context* and *arithmetic coding context* as distinct, unrelated things.

### 3.2.1 HDQ Setting

The problem of quantizer optimization in the HDQ setting under CASQ can still be formulated in the same way shown in (2.7), which we recall here:

$$\inf_{Q} R(Q) \quad \text{s.t.} \quad D(Q) \leq D_T \tag{3.3}$$

where $Q$ is now the overall set of quantization tables described above. The rate $R(Q)$ now denotes the bit rate resulting from compressing an image under CASQ using this new definition of $Q$, and the distortion $D(Q)$ now denotes the corresponding distortion. But given that $Q$ is now a collection of individual quantization tables, then we cannot necessarily apply any of the methods discussed in Section 2.3.1 directly to solve (3.3).

Instead, in light of the discussion of OptD theory in Section 2.3.1, we will take a slightly different approach to quantizer design in the HDQ setting.

The OptD theory from [7] discussed earlier, despite being formulated in the SDQ setting, presents an ideal basis on which we formulate optimal quantizer design in the HDQ setting, for the following reasons. First, OptD is not limited to any particular transform method, entropy coding method, block size or quantization table size; this gives us the freedom to generalize our HDQ problem formulation and solution accordingly. Second, the

problem of extending OptD to fit the multi-context scenario of CASQ is easily solved, as we will see in Section 3.3. Third, it has been shown in [7] that OptD performs superior to previous state-of-the-art methods in the HDQ setting, in both the RD sense and the computational complexity sense. Finally, recall that OptD only produces a quantization table and does not otherwise modify the quantized coefficients in any way, making it a valid HDQ framework despite being formulated in the SDQ setting.

We now present our formulation of the problem of optimal $Q$ design in the HDQ setting. Recall (2.11):

$$\inf_{Q} \inf_{Q'_Q} R_\phi(Q, Q'_Q) \quad \text{s.t.} \quad D(Q, Q'_Q) \leq D_T \tag{3.4}$$

where $Q$ is now the overall quantization table defined earlier, and $Q'_Q$ is now defined as an overall collection of per-contexts mappings:

$$Q'_Q = \left\{ Q'_{Q_0}, Q'_{Q_1}, Q'_{Q_2}, ..., Q'_{Q_{(n_C-1)}} \right\} \tag{3.5}$$

In the above definition, $Q'_{Q_c}$ is a mapping from a sequence of transform coefficients of length $n_c$ at the $i$th frequency position corresponding to the $c$th context to an index sequence of length $n_c$ to be uniformly reconstructed using the quantization table $Q_c$:

$$Q'_{Q_c} : \mathbb{R}^{n_c} \longrightarrow \{0, \pm 1, \pm 2, ...\}^{n_c} \tag{3.6}$$

Equation (3.4) can be converted to an unconstrained optimization using the Lagrange multiplier $\theta$:

$$\inf_{Q} \inf_{Q'_Q} \left[ R_\phi(Q, Q'_Q) + \theta D(Q, Q'_Q) \right] \tag{3.7}$$

where $\theta$ is fixed and represents the rate-distortion trade-off. In practice, $\theta$ corresponds to some bit-rate budget or distortion budget

We then define the optimal quantizer $Q$ in the HDQ setting to be the solution for $Q$ in (3.4) or (3.7) when $Q'_Q$ is intentionally left unsolved. This will be solved in Section 3.3.

### 3.2.2 SDQ Setting

Recall from Section 2.2.2 that the problem of optimal quantizer design in the SDQ setting is tightly coupled to the entropy coder. While the problem in HDQ is generalized to any

entropy coder, we must explicitly define the entropy coder in SDQ. For this purpose, we will consider two entropy coding methods: JPEG-style Huffman coding and ARL coding. The actual implementation of Huffman coding under CASQ differs slightly from that of baseline JPEG; similarly, the actual implementation of ARL coding under CASQ differs slightly from that of [1]. These differences are minor and the reader is assured that the details of these differences are not necessary for following this section of the thesis. These differences are described later in Section 4.4.

We only consider the AC coefficients for optimization under SDQ. Optimization of the DC coefficients under SDQ is typically not explored by the authors of previous works relating to SDQ [6,8–10]. As such, we remove the optimization of the DC coefficients from consideration in this thesis. We assume that DC coefficients are quantized and coded in the same manner as in JPEG (in the Huffman case) or in ARL.

### Huffman Coding

In the Huffman case, we define an AC Huffman table $H_c$ corresponding to the $c$th context, one for each $0 \leq c < n_C$. Assume that $H_c$ will be used for the entropy coding of all run-size sequences associated with the $c$th context. We define $H$ to be the collection of Huffman tables over all contexts:

$$H = \{H_0, H_1, ..., H_{(n_C-1)}\} \tag{3.8}$$

Define $(r, s, id)_c$ as the sequence of run-size pairs $(r, s)$ and quantized indices $id$ associated with the $c$th context. Let $(r, s, id)$ be the sequence of all run-size pairs and quantized index values across all contexts:

$$(r, s, id) = \{(r, s, id)_0, (r, s, id)_1, ..., (r, s, id)_{(n_C-1)}\} \tag{3.9}$$

Let $Q$ be the collection of quantization tables over all contexts as defined earlier. We formulate the problem of optimal quantizer design in the same manner as (2.13):

$$\inf_{(r,s,id),H,Q} R((r, s), H) \quad \text{s.t.} \quad D((r, s, id)_{(Q)}) \leq D_T \tag{3.10}$$

where $R$ now represents the bit rate resulting from compressing the run-size sequences $(r, s)$ using all Huffman tables in $H$ and $D$ represents the corresponding distortion.

As before, this can be converted to an unconstrained optimization via the Lagrange multiplier $\theta$:

$$\inf_{(r,s,id),H,Q} \left[R((r, s), H) + \theta D((r, s, id)_{(Q)})\right] \tag{3.11}$$

Optimal quantization under CASQ in the SDQ setting using Huffman coding then corresponds to the set of $(r, s, id)$, $H$ and $Q$ which minimizes (3.10) or (3.11). Since $(r, s, id)$, $H$ and $Q$ are defined over all contexts, then a full solution for $(r, s, id)$, $H$ and $Q$ implies that the individual $(r, s, id)^c$, $H_c$ and $Q_c$, $0 \leq c < n_C$, are optimized jointly over all contexts.

### ARL Coding

The case of ARL coding is very similar to the Huffman case. Define $(r, l)_c$ as the sequence of run-level pairs corresponding to the $c$th quantization context. Define $(r, l)$ as the collection of all sequences of run-level pairs across all quantization contexts:

$$(r, l) = \{(r, l)_0, (r, l)_1, ..., (r, l)_{(n_C-1)}\} \tag{3.12}$$

As we will see in Section 4.4.2, we define a separate set of arithmetic coding contexts for each quantization context. Define $M_c$ as the collection of arithmetic coding context models corresponding to the $c$th quantization context. Define $M$ as the set of all $M_c$:

$$M = \{M_0, M_1, ..., M_{(n_C-1)}\} \tag{3.13}$$

Then, the problem of quantizer optimization can be written in the same way as (2.21):

$$\inf_{(r,l),M,Q} R((r, l), M) \quad \text{s.t.} \quad D((r, l)_Q) \leq D_T \tag{3.14}$$

where $R$ now denotes the bit rate resulting from compressing the run-length sequences $(r, l)$ from all quantization contexts using the arithmetic coding contexts $M$, and $D$ denotes the corresponding distortion.

This can be converted to an unconstrained optimization problem via the Lagrange multiplier $\lambda$:

$$\min_{(r,l),M,Q} [J(\lambda) = D((r, l)_Q) + \lambda R((r, l), M)] \tag{3.15}$$

Optimal quantization under CASQ in the SDQ setting using ARL coding then corresponds to the set of $(r, l)$ and $Q$ which minimizes (3.14) or (3.15). We do not optimize the arithmetic context selection in $M$, but it remains a parameter upon which the optimal $(r, l)$ and $Q$ depend; rather, the arithmetic coding context selected for coding each run or level value is chosen in the normal manner described in Section 4.4.2. Since $(r, l)$ and $Q$ are defined over all quantization contexts, then a full solution for $(r, l)$ and $Q$ implies that the individual $(r, l)_c$ and $Q_c$ are optimized jointly over all quantization contexts.

## 3.3 OptD with Variable-Length Sources

In this section, we present the details of OptD theory from [7] by describing each step of the solution with the necessary modifications to solve (3.4).

### Modelling

Recall that each transform frequency position $i$, $0 \leq i < L$, can be associated with any context index $c$, $0 \leq c < n_C$. Thus, there are $L' = n_C \cdot L$ possible combinations of transform frequency positions and context indices. Recall as well that there is a separate step size associated with each of these combinations ($Q$ contains $L'$ step sizes). Thus, the step size corresponding to one combination of transform frequency position and context index may be optimized separately from the step size of a combination of the same frequency position and another context index. Under OptD theory, we can simply treat these combinations as separate "frequency positions" for the purpose of optimization of $Q$, indexed by $i$, $0 \leq i < L'$. Thus, from this point on, the notion of having separate contexts is largely absorbed into this new simplified definition of frequency position indexed from 0 to $L' - 1$. This simplifies the syntax of the math that follows.

We begin by modelling the transform coefficients located at each frequency position $i$, $0 \leq i < L'$, as an independent random source $X_i$ with some probability distribution. In [7], each source $X_i$ is assumed to have equal length. However, under CASQ there is no guarantee that the number of blocks associated with one context is equal to that of any other context. Keeping in mind that two different values of $i$ may correspond to two different contexts, then the assumption that each source $X_i$ has equal length no longer holds. Therefore, the first necessary modification to the OptD theory from [7] is to relax this assumption and allow different $X_i$ to have different lengths. We then define $n_i$ to be the length of source $X_i$, and we further define $n_T = \sum_i n_i$ to be the total number of coefficients in $I_0$. Even though all $n_i$ corresponding to one context will be equal (this easily follows from the fact that each block has the same size as every other block, regardless of the contexts), we make no assumptions whatsoever about the relationships between any $n_i$.

With the assumption that the $X_i$ are independent, then we may also assume that the coefficients from any one frequency position can be optimally quantized and encoded separately by quantizing and encoding each $X_i$ separately. We also assume that each $X_i$ has zero mean; this is approximately true in practice. Even if the mean of a source $X_i$ is significantly far from zero, it can simply be subtracted first.

**Optimization**

We now provide explicit definitions of the rate $R_\phi(Q, Q'_Q)$ and distortion $D(Q, Q'_Q)$. In [7], these are defined as the average rate and average distortion, respectively. Let $R_\phi(X_i, q_i, Q'_{q_i})$ denote the average rate, in bits per coefficient, of quantizing and encoding source $X_i$ using step size $q_i$ and quantizer mapping $Q'_{q_i}$ with entropy coding method $\phi$. Let $D(X_i, q_i, Q'_{q_i})$ denote the average distortion per coefficient between $X_i$ and the reconstructed approximation $\hat{X}_i$ given by $q_i$ and $Q'_{q_i}$. Keeping in mind that the sources may have different lengths, then the contribution of $R_\phi(X_i, q_i, Q'_{q_i})$ to the *total* rate (and likewise for the distortion) should be weighted by $n_i$. Thus, we define $R_\phi(Q, Q'_Q)$ as the average rate where the contribution from each source is weighted appropriately. Likewise, we define $D(Q, Q'_Q)$ as the average distortion:

$$R_\phi(Q, Q'_Q) \triangleq \sum_{i=0}^{L'-1} \frac{n_i}{n_T} R_\phi(X_i, q_i, Q'_{q_i}) \tag{3.16}$$

$$D(Q, Q'_Q) \triangleq \sum_{i=0}^{L'-1} \frac{n_i}{n_T} D(X_i, q_i, Q'_{q_i}) \tag{3.17}$$

As in [7], we rewrite (3.7) as

$$\inf_{Q} \inf_{Q'_Q : D(Q,Q'_Q) \le D_T} R_\phi(Q, Q'_Q) \tag{3.18}$$

where $D_T$ is the distortion budget in units of distortion per coefficient.

Under the assumption that the $X_i$ can be quantized and encoded independently, then as in [7] we may split up the inner minimization in (3.18) by optimizing each frequency position separately according to some average per-frequency distortion budget $D_i$, $0 \le i < L'$. These individual distortion budgets $D_i$ are then chosen optimally by an additional outer

minimization. Hence, the inner minimization in (3.18) may be written as

$$\begin{aligned}
\inf_{Q'_Q : D(Q,Q'_Q) \le D_T} R_\phi(Q, Q'_Q) &= \inf_{\substack{\{D_i\}_{i=0}^{L'-1} \\ \sum_{i=0}^{L'-1} n_i D_i = n_T D_T}} \inf_{\substack{Q'_{q_i} : D(X_i, q_i, Q'_{q_i}) \le D_i \\ 0 \le i < L'}} \sum_{i=0}^{L'-1} \frac{n_i}{n_T} R_\phi(X_i, q_i, Q'_{q_i}) \\
&= \inf_{\substack{\{D_i\}_{i=0}^{L'-1} \\ \sum_{i=0}^{L'-1} n_i D_i = n_T D_T}} \sum_{i=0}^{L'-1} \frac{n_i}{n_T} \left[ \inf_{\substack{Q'_{q_i} \\ D(X_i, q_i, Q'_{q_i}) \le D_i}} R_\phi(X_i, q_i, Q'_{q_i}) \right] \\
&= \inf_{\substack{\{D_i\}_{i=0}^{L'-1} \\ \sum_{i=0}^{L'-1} n_i D_i = n_T D_T \\ D_i \ge D(X_i, q_i)}} \sum_{i=0}^{L'-1} \frac{n_i}{n_T} \left[ \inf_{\substack{Q'_{q_i} \\ D(X_i, q_i, Q'_{q_i}) \le D_i}} R_\phi(X_i, q_i, Q'_{q_i}) \right] \qquad (3.19)
\end{aligned}$$

where $D(X_i, q_i)$ is defined as the minimal average distortion per coefficient for $X_i$ that is achievable by uniform reconstruction using $q_i$. From [7], we note that $D(X_i, q_i)$ is equal to the average distortion from the HDQ of $X_i$ with quantization step size $q_i$. Also from [7], the last equality in (3.19) follows from the fact that if $D_i < D(X_i, q_i)$, then there does not exist a quantizer such that $D(X_i, q_i, Q'_{q_i}) \le D_i$ (since $D(X_i, q_i)$ was defined as the *minimal achievable* distortion).

The next step from [7] uses universal redundancy results from lossy source coding theory [11], which say that when $\phi$ is universal and optimal, then

$$\inf_{\substack{Q'_{q_i} \\ D(X_i, q_i, Q'_{q_i}) \le D_i}} R_\phi(X_i, q_i, Q'_{q_i}) = R_{X_i}^{q_i}(D_i) + K \left( 1 + 2 \left\lfloor 1 + \frac{A}{q_i} \right\rfloor \right) \frac{\ln n_i}{n_i} + o\left( \frac{\ln n_i}{n_i} \right) \qquad (3.20)$$

where $R_{X_i}^{q_i}(D_i)$ is the Shannon rate distortion function of $X_i$ with respect to the reconstruction alphabet $\left\{ 0, \pm 1 q_i, \pm 2 q_i, ..., \pm \left\lfloor 1 + \frac{A}{q_i} \right\rfloor q_i \right\}$, $A$ is the largest possible magnitude that a transform coefficient could have, and $K = O(1)$ is some positive bounded term. We can simplify this by noting that $1 + 2 \left\lfloor 1 + \frac{A}{q_i} \right\rfloor$ is proportional to $1/q_i$ and absorbing it into $K$, which leaves us with

$$\inf_{\substack{Q'_{q_i} \\ D(X_i, q_i, Q'_{q_i}) \le D_i}} R_\phi(X_i, q_i, Q'_{q_i}) = R_{X_i}^{q_i}(D_i) + \frac{K}{q_i} \frac{\ln n_i}{n_i} + o\left( \frac{\ln n_i}{n_i} \right) \qquad (3.21)$$

Substituting (3.21) back into (3.19) and (3.18), we now have

$$\begin{aligned}
&\inf_{Q} \inf_{Q'_Q : D(Q,Q'_Q) \le D_T} R_\phi(Q, Q'_Q) \\
&\quad = \inf_{Q} \inf_{\substack{\{D_i\}_{i=0}^{L'-1} \\ \sum_{i=0}^{L'-1} n_i D_i = n_T D_T \\ D_i \ge D(X_i, q_i)}} \sum_{i=0}^{L'-1} \frac{n_i}{n_T} \left[ R_{X_i}^{q_i}(D_i) + \frac{K}{q_i} \frac{\ln n_i}{n_i} + o\left( \frac{\ln n_i}{n_i} \right) \right] \\
&\quad = \inf_{Q} \inf_{\substack{\{D_i\}_{i=0}^{L'-1} \\ \sum_{i=0}^{L'-1} n_i D_i = n_T D_T \\ D_i \ge D(X_i, q_i)}} \sum_{i=0}^{L'-1} \left[ \frac{n_i}{n_T} R_{X_i}^{q_i}(D_i) + \frac{K}{q_i} \frac{\ln n_i}{n_T} + o\left( \frac{\ln n_i}{n_T} \right) \right] \qquad (3.22)
\end{aligned}$$

The next step in [7] is to further lower bound $R_{X_i}^{q_i}(D_i)$ by the Shannon lower bound to the rate distortion function of $X_i$, because there is no analytic formula for $R_{X_i}^{q_i}(D_i)$ in general. This lower bound is given by $R_{X_i}^{(SL)}(D_i)$:

$$\begin{aligned}
R_{X_i}^{q_i}(D_i) &\ge R_{X_i}^{(SL)}(D_i) \\
&= \max\left\{ H(X_i) - \frac{1}{2} \log 2\pi e D_i, 0 \right\} \\
&= \begin{cases} H(X_i) - \frac{1}{2} \log 2\pi e D_i & \text{if } \hat{\sigma}_i^2 > D_i \\ 0 & \text{otherwise} \end{cases} \qquad (3.23)
\end{aligned}$$

where $H(X_i)$ is the differential entropy of $X_i$, and $\hat{\sigma}_i^2$ is chosen such that $H(X_i) = \frac{1}{2} \log 2\pi e \hat{\sigma}_i^2$ (assume $\log(\cdot)$ is base 2).

Substituting into (3.22), we have

$$\begin{aligned}
&\inf_{Q} \inf_{Q'_Q : D(Q,Q'_Q) \le D_T} R_\phi(Q, Q'_Q) \\
&\quad \ge \inf_{Q} \inf_{\substack{\{D_i\}_{i=0}^{L'-1} \\ \sum_{i=0}^{L'-1} n_i D_i = n_T D_T \\ D_i \ge D(X_i, q_i)}} \sum_{i=0}^{L'-1} \left[ \frac{n_i}{n_T} R_{X_i}^{(SL)}(D_i) + \frac{K}{q_i} \frac{\ln n_i}{n_T} + o\left( \frac{\ln n_i}{n_T} \right) \right] \qquad (3.24)
\end{aligned}$$

In the above equation, we can further limit $Q$ to only those which satisfy

$$\sum_{i=0}^{L'-1} n_i D(X_i, q_i) \le n_T D_T \qquad (3.25)$$

since there exists no choice of $\{D_i\}$ that satisfies $\sum_i n_i D_i = n_T D_T$ with $D_i \geq D(X_i, q_i)$ if (3.25) is not satisfied.

Suppose that $Q$ is now given (but is not necessarily optimal) and satisfies (3.25). We are now interested in solving for the optimal $\{D_i\}$ from the inner minimization of (3.24) given this $Q$:

$$\inf_{Q} \inf_{\substack{\{D_i\}_{i=0}^{L'-1} \\ \sum_{i=0}^{L'-1} n_i D_i = n_T D_T \\ D_i \geq D(X_i, q_i)}} \sum_{i=0}^{L'-1} \left[ \frac{n_i}{n_T} R_{X_i}^{(SL)}(D_i) + \frac{K}{q_i} \frac{\ln n_i}{n_T} + o\left( \frac{\ln n_i}{n_T} \right) \right] \tag{3.26}$$

Note that the last two terms do not depend on $D_i$; hence, solving the inner minimization in (3.26) is equivalent to solving

$$\inf_{\substack{\{D_i\}_{i=0}^{L'-1} \\ \sum_{i=0}^{L'-1} n_i D_i = n_T D_T \\ D_i \geq D(X_i, q_i)}} \sum_{i=0}^{L'-1} \left[ \frac{n_i}{n_T} R_{X_i}^{(SL)}(D_i) \right] \tag{3.27}$$

The next step in [7] solves the above minimization when all $n_i$ are equal. However, since we cannot assume that they are equal under CASQ, then we must re-solve the above minimization accordingly.

Since $R_{X_i}^{(SL)}(D_i)$ is a convex function of $D_i$, then there is a neat closed-form solution which can be found using the Karush-Kuhn-Tucker (KKT) conditions.

Let $\vec{D} = [D_0, D_1, ..., D_{(L'-1)}]$ be a row vector of the per-context distortion budgets. Define the objective function $f(\vec{D})$ (we multiply out the constant $n_T$ for simplicity, as it will not affect the solution):

$$f(\vec{D}) = \sum_{i=0}^{L'-1} n_i R_{X_i}^{(SL)}(D_i) \tag{3.28}$$

Define $g_i(\vec{D})$ to be an inequality constraint function, $0 \leq i < L'$:

$$g_i(\vec{D}) = D(X_i, q_i) - D_i \tag{3.29}$$

Define $h(\vec{D})$ to be an equality constraint function:

$$h(\vec{D}) = \sum_{i=0}^{L'-1} n_i D_i - n_T D_T \tag{3.30}$$

Note that all of $f(\cdot)$, $g_i(\cdot)$ and $h(\cdot)$ are convex (recall from (3.23) that $R_{X_i}^{(SL)}(D_i)$ is convex). Suppose that $f(\vec{D})$, $g_i(\vec{D})$ and $h(\vec{D})$ are all differentiable at a point $\vec{D}$. Let $\{\mu_i\}$ and $\lambda$, $0 \leq i < L'$, be the KKT multipliers. We now investigate the KKT conditions for $\vec{D}$ to be an optimum.

*Stationarity:* For minimization of $f(\vec{D})$, the following must hold:

$$-\nabla f(\vec{D}) = \sum_{i=0}^{L'-1} \mu_i \nabla g_i(\vec{D}) + \lambda \nabla h(\vec{D}) \tag{3.31}$$

For now, assume that $D_i < \hat{\sigma}_i^2$, $0 \leq i < L'$. Expanding the left-hand side,

$$\begin{aligned} -\left(\nabla f(\vec{D})\right)_i &= -\sum_{j=0}^{L'-1} n_j \frac{\mathrm{d}}{\mathrm{d}D_i} R_{X_j}^{(SL)}(D_j) \\ &= -n_i \frac{\mathrm{d}}{\mathrm{d}D_i} R_{X_i}^{(SL)}(D_i) \\ &= -n_i \frac{\mathrm{d}}{\mathrm{d}D_i} H(X_i) + \frac{n_i}{2} \frac{\mathrm{d}}{\mathrm{d}D_i} \log 2\pi e D_i \\ &= \frac{n_i}{D_i \cdot 2 \ln 2} \end{aligned} \tag{3.32}$$

where $(\nabla \cdot)_i$ denotes the $i$th entry of the gradient.

Expanding the right-hand side of (3.31),

$$\begin{aligned} &\left( \sum_{j=0}^{L'-1} \mu_j \nabla g_j(\vec{D}) + \lambda \nabla h(\vec{D}) \right)_i \\ &\quad = \sum_{j=0}^{L'-1} \mu_j \frac{\mathrm{d}}{\mathrm{d}D_i} \left( D(X_j, q_j) - D_j \right) + \lambda \frac{\mathrm{d}}{\mathrm{d}D_i} \left( \sum_{j=0}^{L'-1} n_j D_j - n_T D_T \right) \\ &\quad = \lambda n_i - \mu_i \end{aligned} \tag{3.33}$$

Combining (3.33) with (3.31),

$$\begin{aligned}\lambda n_i - \mu_i &= \frac{n_i}{D_i \cdot 2\ln 2} \\ \mu_i &= \lambda n_i - \frac{n_i}{D_i \cdot 2\ln 2}, \quad 0 \le i < L'\end{aligned} \tag{3.34}$$

*Dual feasibility:* The dual feasibility condition from KKT requires $\mu_i \ge 0$, $0 \le i < L'$. We use this to turn the above equation into an inequality (assuming $n_i > 0$):

$$\begin{aligned}\lambda n_i &\ge \frac{n_i}{D_i \cdot 2\ln 2} \\ \lambda &\ge \frac{1}{D_i \cdot 2\ln 2}, \quad 0 \le i < L'\end{aligned} \tag{3.35}$$

*Complementary slackness:* The following must hold for $\vec{D}$ to be an optimum:

$$\mu_i g_i(\vec{D}) = 0, \quad 0 \le i < L' \tag{3.36}$$

Expanding,

$$\mu_i g_i(\vec{D}) = \mu_i (D(X_i, q_i) - D_i) = 0 \tag{3.37}$$

Substituting from (3.34),

$$\begin{aligned}\left(\lambda n_i - \frac{n_i}{D_i \cdot 2\ln 2}\right)(D(X_i, q_i) - D_i) &= 0 \\ \left(\lambda - \frac{1}{D_i \cdot 2\ln 2}\right)(D(X_i, q_i) - D_i) &= 0, \quad 0 \le i < L'\end{aligned} \tag{3.38}$$

Suppose that $\lambda < 1/(D(X_i, q_i)2\ln 2)$. From (3.35), we have

$$\begin{aligned}\frac{1}{D(X_i, q_i)2\ln 2} > \lambda &\ge \frac{1}{D_i 2\ln 2} \\ \frac{1}{D(X_i, q_i)} &> \frac{1}{D_i} \\ D_i &> D(X_i, q_i)\end{aligned} \tag{3.39}$$

Hence, the term $(D(X_i, q_i) - D_i)$ in (3.38) is strictly non-zero and may be cancelled out, leaving

$$\lambda - \frac{1}{D_i \cdot 2 \ln 2} = 0$$
$$D_i = \frac{1}{\lambda 2 \ln 2} \triangleq d \tag{3.40}$$

Note that from the above definition of $d$, the condition $\lambda < 1/(D(X_i, q_i)2 \ln 2)$ is equivalent to $D(X_i, q_i) < d$.

On the other hand, suppose now that $\lambda \geq 1/(D(X_i, q_i)2 \ln 2)$. There are two cases: either $\mu_i = 0$ or $\mu_i > 0$. In the latter case, from (3.37) we see clearly that $D_i = D(X_i, q_i)$. In the former case, the inequality in (3.35) becomes an equality, implying

$$\lambda = \frac{1}{D_i 2 \ln 2} \geq \frac{1}{D(X_i, q_i) 2 \ln 2}$$
$$D_i \leq D(X_i, q_i) \tag{3.41}$$

But the primal conditions of the problem state that $D_i \geq D(X_i, q_i)$; therefore, $D_i = D(X_i, q_i)$ necessarily.

Putting all of this together, we have

$$D_i = \begin{cases} d & \text{if } d > D(X_i, q_i) \\ D(X_i, q_i) & \text{otherwise} \end{cases} \tag{3.42}$$

where $d$ is a constant, referred to as the "water level" in [7], chosen such that $\sum_i n_i D_i = n_T D_T$.

Now we address the previous assumption that $D_i < \hat{\sigma}_i^2$, $0 \leq i < L'$. Suppose that a particular $D_i$ instead meets or exceeds $\hat{\sigma}_i^2$. In this case, no matter what the value of $D_i$ is, the rate contribution from coding source $X_i$ is zero (see (3.23)). There would be no point in allowing $D_i$ to increase beyond $\hat{\sigma}_i^2$ for any $i$: if we did allow this, it would cause at least one $D_j$, $j \neq i$ to decrease due to the constraint $\sum_i n_i D_i = n_T D_T$. This leads to a corresponding increase in rate contribution for coding source $X_j$, and thus an increase in the objective function (3.28). In other words, a more optimal solution can be found if $D_i > \hat{\sigma}_i^2$ by forcing $D_i = \hat{\sigma}_i^2$, since the rate contribution from coding source $X_i$ is zero either way. Therefore, $\hat{\sigma}_i^2$ is an upper bound on $D_i$ if $\{D_i\}$ is optimal. This leads to our

final solution to (3.27):

$$D_i = D_i(Q) = \begin{cases} D(X_i, q_i) & \text{if } d \leq D(X_i, q_i) \\ d & \text{if } D(X_i, q_i) < d \leq \hat{\sigma}_i^2 \\ \hat{\sigma}_i^2 & d > \hat{\sigma}_i^2 \end{cases} \tag{3.43}$$

where $d$ is chosen such that $\sum_i n_i D_i = n_T D_T$. Interestingly, the expression for $D_i$ is identical to the one found in [7]; the only difference is in the selection of $d$.

Substituting (3.43) back into (3.26),

$$\begin{aligned} &\inf_Q \inf_{\substack{\{D_i\}_{i=0}^{L'-1} \\ \sum_{i=0}^{L'-1} n_i D_i = n_T D_T \\ D_i \geq D(X_i, q_i)}} \sum_{i=0}^{L'-1} \left[ \frac{n_i}{n_T} R_{X_i}^{(SL)}(D_i) + \frac{K}{q_i} \frac{\ln n_i}{n_T} + o\left( \frac{\ln n_i}{n_T} \right) \right] \\ &\quad = \inf_Q \sum_{i=0}^{L'-1} \left[ \frac{n_i}{n_T} R_{X_i}^{(SL)}(D_i(Q)) + \frac{K}{q_i} \frac{\ln n_i}{n_T} + o\left( \frac{\ln n_i}{n_T} \right) \right] \\ &\quad = \inf_Q \left[ \sum_{i=0}^{L'-1} \frac{n_i}{n_T} R_{X_i}^{(SL)}(D_i(Q)) + \sum_{i=0}^{L'-1} \left( \frac{K}{q_i} \frac{\ln n_i}{n_T} + o\left( \frac{\ln n_i}{n_T} \right) \right) \right] \end{aligned} \tag{3.44}$$

The next step in [7] is to consider the minimization of the first summation:

$$\inf_Q \sum_{i=0}^{L'-1} \frac{n_i}{n_T} R_{X_i}^{(SL)}(D_i(Q)) \tag{3.45}$$

If we go back a step, we can rewrite the above as

$$\inf_Q \sum_{i=0}^{L'-1} \frac{n_i}{n_T} R_{X_i}^{(SL)}(D_i(Q)) = \inf_Q \inf_{\substack{\{D_i\}_{i=0}^{L'-1} \\ \sum_{i=0}^{L'-1} n_i D_i = n_T D_T \\ D_i \geq D(X_i, q_i)}} \sum_{i=0}^{L'-1} \left[ \frac{n_i}{n_T} R_{X_i}^{(SL)}(D_i) \right] \tag{3.46}$$

We now know from (3.43) that having a fixed $Q$ in the above inner minimization is equivalent to placing a lower bound on each $D_i$, where that lower bound is the minimal achievable distortion from using HDQ on $X_i$ with step size $q_i$, $0 \leq i < L'$. Therefore, it is

not hard to see that solving the above inner minimization over all possible $Q$ is equivalent to removing the lower bound on each $D_i$. Hence,

$$\inf_{Q} \inf_{\substack{\{D_i\}_{i=0}^{L'-1} \\ \sum_{i=0}^{L'-1} n_i D_i = n_T D_T \\ D_i \geq D(X_i, q_i)}} \sum_{i=0}^{L'-1} \left[ \frac{n_i}{n_T} R_{X_i}^{(SL)}(D_i) \right] = \inf_{\substack{\{D_i\}_{i=0}^{L'-1} \\ \sum_{i=0}^{L'-1} n_i D_i = n_T D_T}} \sum_{i=0}^{L'-1} \left[ \frac{n_i}{n_T} R_{X_i}^{(SL)}(D_i) \right] \tag{3.47}$$

This optimization problem is similar to the one solved earlier, except now we omit the inequality constraint functions $g_i(\vec{D})$ and the solution simplifies from the KKT conditions to the Lagrange conditions (and $\lambda$ becomes a Lagrange multiplier).

Specifically, we are now led to solving (for some optimum point $\vec{D}^*$)

$$-\nabla f(\vec{D}^*) = \lambda \nabla h(\vec{D}^*) \tag{3.48}$$

Similar to before, we find (for $D_i^* < \hat{\sigma}_i^2$)

$$\begin{aligned} -\left(\nabla f(\vec{D}^*)\right)_i &= \frac{n_i}{D_i^* \cdot 2 \ln 2} \\ \lambda \left(\nabla h(\vec{D}^*)\right) &= \lambda n_i \\ \implies D_i^* &= \frac{1}{\lambda 2 \ln 2} \triangleq d^* \end{aligned} \tag{3.49}$$

By the same argument as before, we upper bound $D_i^*$ by $\hat{\sigma}_i^2$, giving the solution

$$D_i^* = \begin{cases} d^* & \text{if } d^* < \hat{\sigma}_i^2 \\ \hat{\sigma}_i^2 & \text{otherwise} \end{cases} \tag{3.50}$$

where $d^*$ is chosen such that

$$\sum_{i=0}^{L'-1} n_i D_i^* = n_T D_T \tag{3.51}$$

Again, the expression for $D_i^*$ above is identical to the one found in [7]; the only difference is in the selection of $d^*$.

By the next step in [7], we now define a quantization table $Q^* = \{q_0^*, q_1^*, ..., q_{(L'-1)}^*\}$ where

$$q_i^* = \sup\{q_i : D(X_i, q_i) \leq D_i^*\}, \quad 0 \leq i < L' \tag{3.52}$$

Assume that $D(X_i, q_i)$ is a strictly increasing and differentiable function of $q_i$ (this holds for the mean-squared error distortion metric). Then from the above definition of $q_i^*$, clearly

$$D(X_i, q_i^*) = D_i^* \tag{3.53}$$

$$D_i(Q^*) = D_i^* \tag{3.54}$$

Since $D_i(Q^*) = D_i^*$, it is implied that $Q^*$ is the optimum of (3.45).

We note that among all $Q$ satisfying (3.25) and $D_i(Q) = D_i^*$, $Q^*$ is the largest in the sense that the following holds:

$$\delta(Q) \triangleq \max\{q_i - q_i^* : 0 \leq i < L'\} \leq 0 \tag{3.55}$$

We now draw a connection between $Q^*$ and the optimal quantization table to (3.44). Define $Q^0$ as the optimal quantization table to (3.44). Note that the first summation in (3.44) is a non-decreasing function of each $q_i$, while the second summation is a strictly decreasing function of each $q_i$. In some sense, the second summation represents the "quality cost" of increasing $q_i$ and facilitates the rate-distortion trade-off in choosing the optimal $Q$. However, we note that as $n_i$ grows large for every $i$, the first summation will dominate. Therefore, when $n_i$ is large for every $i$, we expect $Q^*$ to be very close to the actual optimal $Q^*$. We formalize and prove this in the following theorem.

**Theorem 1.** *Assume that $D(X_i, q_i)$, $0 \leq i < L'$, is a strictly increasing and differentiable function of $q_i$. Assume that each source $X_i$ has a corresponding length $n_i$. Define $n_m = \min_i n_i$. Then the optimal $Q^0 = (q_0^0, q_1^0, ..., q_{(L'-1)}^0)$ satisfies*

$$\left|Q^0 - Q^*\right| = O\left(\frac{\log n_m}{n_m}\right) \tag{3.56}$$

*where $Q^* = (q_0^*, q_1^*, ..., q_{(L'-1)}^*)$ is defined via (3.52) and $|\cdot|$ denotes the Euclidean distance.*

*Proof.* The proof largely follows that of Theorem 1 in [7], with some minor details changed. We reproduce the proof here for completeness.

For any $Q$ satisfying (3.25) (but not necessarily satisfying $D_i(Q) = D_i^*$), we define

$$F(Q) \triangleq \sum_{i=0}^{L'-1} \frac{n_i}{n_T} R_{X_i}^{(SL)}(D_i(Q)) + \sum_{i=0}^{L'-1} \left(\frac{K}{q_i}\frac{\ln n_i}{n_T} + \frac{o(\ln n_i)}{n_T}\right) \tag{3.57}$$

$$D(Q) \triangleq (D_0(Q), D_1(Q), ..., D_{(L'-1)}(Q)) \tag{3.58}$$

We want to compare $F(Q)$ with $F(Q^*)$. We define four different cases:

(i) $\delta(Q) < 0$

(ii) $\delta(Q) \geq C_1\sqrt{\frac{\ln n_M}{n_M}}$

(iii) $C_2\frac{\ln n_m}{n_m} \leq \delta(Q) < C_1\sqrt{\frac{\ln n_M}{n_M}}$

(iv) $0 \leq \delta(Q) < C_2\frac{\ln n_m}{n_m}$

where $C_1 > 0$ and $C_2 > 0$ are constants to be discussed later, and $n_M \triangleq \max_i\{n_i\}$. Assume that $n_m = O(n_M)$, which is apparent from the definitions of $n_m$ and $n_M$ when they are both large. We will work towards placing $Q^0$ into one of the cases above, the bounds of which will imply a bound on $|Q^0 - Q^*|$.

*Case (i)*: From (3.52) and (3.55), it should be clear that if $\delta(Q) < 0$ then $D_i(Q) = D_i^* = D_i(Q^*)$, $0 \leq i < L'$. Therefore, the first summation in $F(Q)$ is equal to that of $F(Q^*)$. Furthermore, since $\delta(Q)$ is strictly less than zero, it is implied that $q_i < q_i^*$ is also strict for every $i$. Consequently, the second summation in $F(Q)$ is strictly greater than that of $F(Q^*)$, and thus $F(Q) > F(Q^*)$. Since $F(Q)$ is exactly the RD cost function being minimized over $Q$ in (3.44), then $Q$ cannot possibly be optimal. We conclude that $Q^0$ cannot fall into this case.

*Case (ii)*: Recall that $\delta(Q)$ denotes the largest value of $q_i - q_i^*$ over all $i$. In this case, it is guaranteed that for at least one $i$, $q_i - q_i^* \geq C_1\sqrt{\frac{\ln n_M}{n_M}}$. Recall the assumption that $D(X_i, q_i)$ is a strictly increasing function of $q_i$; then, it is also true that $D(X_i, q_i) - D(X_i, q_i^*) \geq \hat{C}_1\sqrt{\frac{\ln n_M}{n_M}}$ for some positive constant $\hat{C}_1$, for the same $i$ as previously mentioned. Since $D(X_i, q_i)$ is a lower bound to $D_i(Q)$ (see (3.43)), it is further implied that $D_i(Q) - D_i(Q^*) \geq \hat{C}_1\sqrt{\frac{\ln n_M}{n_M}}$ as well. Therefore,

$$|D(Q) - D(Q^*)| \geq c_1 C_1\sqrt{\frac{\ln n_M}{n_M}} \tag{3.59}$$

for some positive constant $c_1$.

Next, define the function

$$G(D(Q)) = \sum_{i=0}^{L'-1} \frac{n_i}{n_T} R_{X_i}^{(SL)}(D_i(Q)) \tag{3.60}$$

Note that $G(D(Q))$ is the same as the objective function from (3.45), for which $Q^*$ is a strict minimum. This implies that the gradient of $G(D(Q))$ evaluated at $Q^*$ is zero:

$$\nabla G(D(Q^*)) = \vec{0} \tag{3.61}$$

Let $A \triangleq D(Q) - D(Q^*)$. If we take the Taylor expansion of $G(D(Q))$ around $Q^*$,

$$\begin{aligned} G(D(Q)) &= G(D(Q^*)) + \nabla G(D(Q^*)) \cdot A + \frac{1}{2} A \cdot \nabla^2 G(D(\hat{Q})) A^T \\ &= G(D(Q^*)) + \frac{1}{2} A \cdot \nabla^2 G(D(\hat{Q})) A^T \end{aligned} \tag{3.62}$$

for some point $D(\hat{Q})$ in the neighbourhood of $D(Q^*)$. Since $Q^*$ is a strict minimum of $G(D(Q))$, then necessarily $G(D(Q)) > G(D(Q^*))$ for any $Q \neq Q^*$, implying that the Hessian must be positive definite at $D(\hat{Q})$. From the differentiable property of $D(X_i, q_i)$, it follows that the Hessian is also symmetric, meaning there exists a Cholesky decomposition

$$\nabla^2 G(D(\hat{Q})) = LL^T \tag{3.63}$$

where $L$ is a lower triangular $L' \times L'$ matrix. Hence,

$$\begin{aligned} G(D(Q)) &> G(D(Q^*)) + \frac{1}{2}(AL)(AL)^T \\ &= G(D(Q^*)) + \frac{1}{2}|AL|^2 \\ &= G(D(Q^*)) + C|D(Q) - D(Q^*)|^2 \end{aligned} \tag{3.64}$$

for some positive constant $C$.

We therefore have

$$\sum_{i=0}^{L'-1} \frac{n_i}{n_T} R_{X_i}^{(SL)}(D_i(Q)) > \sum_{i=0}^{L'-1} \frac{n_i}{n_T} R_{X_i}^{(SL)}(D_i(Q^*)) + \hat{c}_1 c_1^2 C_1^2 \frac{\ln n_M}{n_M} \tag{3.65}$$

for some positive constant $\hat{c}_1$. Hence,

$$\begin{aligned} &F(Q) - F(Q^*) \\ &\quad > \hat{c}_1 c_1^2 C_1^2 \frac{\ln n_M}{n_M} + \sum_{i=0}^{L'-1} \frac{K}{q_i} \frac{\ln n_i}{n_T} - \sum_{i=0}^{L'-1} \frac{K}{q_i^*} \frac{\ln n_i}{n_T} + \sum_{i=0}^{L'-1} \frac{o(\ln n_i)}{n_T} \\ &\quad > \hat{c}_1 c_1^2 C_1^2 \frac{\ln n_M}{n_M} - \sum_{i:q_i<q_i^*} \frac{K}{q_i^*} \frac{\ln n_i}{n_T} + \sum_{i=0}^{L'-1} \frac{o(\ln n_i)}{n_T} \\ &\quad \geq \hat{c}_1 c_1^2 C_1^2 \frac{\ln n_M}{n_M} - \sum_{i:q_i<q_i^*} \frac{K}{q_i^*} \frac{\ln n_M}{n_M} + \sum_{i=0}^{L'-1} \frac{o(\ln n_i)}{n_T} > 0 \end{aligned} \tag{3.66}$$

for large $n_m$ and $n_M$, and when $C_1$ is chosen properly. The last line follows from the fact that $n_M = O(n_T)$ and $n_i = O(n_M)$, $0 \le i < L'$. Therefore, $F(Q) > F(Q^*)$, and $Q^0$ cannot fall into this case.

*Case (iii)*: Similarly to Case (ii), we have

$$\begin{aligned} &F(Q) - F(Q^*) \\ &\quad > \hat{c}_1 |D(Q) - D(Q^*)|^2 - \sum_{i:q_i > q_i^*} \left( \frac{K}{q_i^*} - \frac{K}{q_i} \right) \frac{\ln n_i}{n_T} + \sum_{i=0}^{L'-1} \frac{o(\ln n_i)}{n_T} \end{aligned} \tag{3.67}$$

From the same argument in Case (ii), we know that $c_2 C_2 \frac{\ln n_m}{n_m} \le |D(Q) - D(Q^*)| < c_1 C_1 \sqrt{\frac{\ln n_M}{n_M}}$. Therefore, if we write $|D(Q) - D(Q^*)|^2$ as $|D(Q) - D(Q^*)| \cdot |D(Q) - D(Q^*)|$, then we can replace one of the terms with $\hat{c}_3 \delta(Q)$, $\hat{c}_3 > 0$ to establish an asymptotic lower bound since $C_2 \frac{\ln n_m}{n_m} \le \delta(Q) < C_1 \sqrt{\frac{\ln n_M}{n_M}}$:

$$\begin{aligned} &F(Q) - F(Q^*) \\ &\quad > \hat{c}_3 \delta(Q) |D(Q) - D(Q^*)| - \sum_{i:q_i > q_i^*} \left( \frac{K}{q_i^*} - \frac{K}{q_i} \right) \frac{\ln n_i}{n_T} + \sum_{i=0}^{L'-1} \frac{o(\ln n_i)}{n_T} \end{aligned} \tag{3.68}$$

Note that for large $n_M$, the bounds on $\delta(Q)$ imply that $q_i$ is very close to $q_i^*$ when $q_i > q_i^*$. Therefore, we can approximate $1/q_i^* - 1/q_i$ by its tangent at $q_i^*$. Noting that $q_i^*$ is fixed, we get

$$\frac{1}{q_i^*} - \frac{1}{q_i} \approx \hat{c}_4 (q_i - q_i^*) \approx \hat{c}_4 \delta(Q) \tag{3.69}$$

for some positive constant $\hat{c}_4$. Therefore,

$$\begin{aligned} &F(Q) - F(Q^*) \\ &\quad > \hat{c}_3 \delta(Q) \left[ |D(Q) - D(Q^*)| - c_4 \frac{\ln n_i}{n_T} \right] + \sum_{i=0}^{L'-1} \frac{o(\ln n_i)}{n_T} \end{aligned} \tag{3.70}$$

for some positive constant $c_4$. From the lower bound on $|D(Q) - D(Q^*)|$, we then have

$$\begin{aligned} & F(Q) - F(Q^*) \\ & \quad > c_3\delta(Q)\left[c_2C_2\frac{\ln n_m}{n_m} - \frac{\ln n_i}{n_T}\right] + \sum_{i=0}^{L'-1}\frac{o(\ln n_i)}{n_T} \\ & \quad \geq c_3\delta(Q)\left[c_2C_2\frac{\ln n_m}{n_m} - \frac{\ln n_M}{n_M}\right] + \sum_{i=0}^{L'-1}\frac{o(\ln n_i)}{n_T} \end{aligned} \tag{3.71}$$

for some positive constant $c_3$. Note that since $n_m = O(n_M)$ and since $\ln x/x$ decays towards zero for large $x$, then $\ln n_m/n_m$ is an upper bound on $\ln n_M/n_M$ for large $n_m$. Therefore,

$$\begin{aligned} & F(Q) - F(Q^*) \\ & \quad > c_3\delta(Q)\left[c_2C_2\frac{\ln n_M}{n_M} - \frac{\ln n_M}{n_M}\right] + \sum_{i=0}^{L'-1}\frac{o(\ln n_i)}{n_T} > 0 \end{aligned} \tag{3.72}$$

for large $n_m$.

This implies that $Q^0$ cannot fall into Case (iii). Given that it cannot fall into Case (i) or Case (ii) either, it must then fall into Case (iv), where

$$0 \leq \delta(Q^0) < C_2\frac{\ln n_m}{n_m} \tag{3.73}$$

This, together with (3.43) and (3.50) implies that $q_i^0 \geq q_i^* - O(\frac{\ln n_m}{n_m})$ for any $q_i^0 < q_i^*$, therefore

$$|Q^0 - Q^*| = O\left(\frac{\ln n_m}{n_m}\right) \tag{3.74}$$

□

*Remark.* Note that this is a slightly weaker version of Theorem 1 in [7]. The weakness comes from the fact that even if $n_T$ grows large, there is no guarantee that each $n_i$ will also grow large with $n_T$; consequently, for any $i$ for which $n_i = o(n_T)$, it cannot be shown that the corresponding $q_i^*$ will always converge to $q_i^*$ for large $n_T$. Nonetheless, this weakness is not significant in practice for two reasons. First, for any reasonable context selection method for the blocks of an image (including the classifier proposed in Section 4.2.2), the number of blocks associated with each context should grow roughly linearly with the overall size of the

image, implying that $n_i = \Theta(n_T)$ for all $i$ and, therefore, that $|Q^0 - Q^*| = O\left(\log(n_T)/n_T\right)$. Second, and more generally: although $Q^*$ may not converge to $Q^0$ in general when $n_T$ is large, it is intuitive that the RD performance under $Q^*$ will nonetheless converge in general to the RD performance under $Q^0$ because the *relative* RD contribution from any source $X_i$ for which $n_i = o(n_T)$ will become negligible as $n_T$ grows large. This follows from the fact that in (3.16) and (3.17), the RD contribution from source $X_i$ is weighted by $n_i$, $0 \leq i < L'$. As $n_T$ grows large, the contributions to overall rate and distortion from the sources where $n_i = \Theta(n_T)$ will dominate the contributions from the sources where $n_i = o(n_T)$, and thus the sub-optimality of $q_i$ for those sources being dominated will have negligible impact on the overall RD performance.

With this in mind, we can compute $Q^*$ and be assured by Theorem 1 and the remarks above that the RD performance under $Q^*$ will be close to the RD performance under $Q^0$ when $n_T$ is large; in other words, the RD performance under $Q^*$ is nearly optimal for sufficiently large images. In the following sub-section, we will apply our modified OptD theory to DCT-based image coding under CASQ.

### Application to DCT-based Image Coding under CASQ

In [7], an efficient algorithm is proposed to compute $Q^*$ for a DCT-based image coding system, where the distortion function $D(X_i, q_i)$ is approximated based on modelling each source $X_i$ as a certain distribution. Despite the lengthy re-derivation of OptD theory under CASQ, this algorithm applies directly to CASQ without modification. The only considerations that need to be made are for the calculation of the "water level" $d$ and for the fact that, when there are multiple contexts, then there may also be multiple DC frequencies; accordingly, care should be taken in modelling the random sources corresponding to those frequencies, because the DC models are chosen differently from the AC models. We will re-produce the algorithm here for completeness but we omit the details of its derivation, which are completely unchanged from [7]. Assume that the source variance $\sigma_i^2$ is used in place of $\hat{\sigma}_i^2$; the purpose of this is to simplify the computation of $Q^*$ without having to compute the differential entropy of each $X_i$.

In the case of a DC frequency when only one context is used ($i = 0$), the source is simply modelled as a uniform random sequence, and thus we have $D(X_0, q_0) = q_0^2/12$. In the case of AC frequencies, each source is modelled as a Laplacian source with parameter $\lambda_i$. The probability density function $f_i(x)$ is given by

$$f_i(x) = \frac{1}{2\lambda_i} e^{-\frac{|x|}{\lambda_i}} \tag{3.75}$$

$D(X_i, q_i)$ for the AC frequency positions is then approximated by the distortion of a dead-zone quantizer with uniform reconstruction, denoted by $D_{Lap}(\lambda_i, q_i)$, $1 \leq i < L'$:

$$D_{Lap}(\lambda_i, q_i) \triangleq 2\lambda_i^2 - \frac{2q_i(\lambda_i + s_i - 0.5q_i)}{e^{s_i/\lambda_i}(1 - e^{-q_i/\lambda_i})} \tag{3.76}$$

where the dead-zone size $s_i$ is

$$s_i = q_i - \lambda_i + \frac{q_i}{e^{q_i/\lambda_i} - 1} \tag{3.77}$$

The derivations for (3.76) and (3.77) may be found in [12]. Note that the Laplacian parameters $\{\lambda_i\}$ may be efficiently estimated via maximum likelihood from the sample values $x_i(0), x_i(1), ..., x_i(n_i - 1)$ of $X_i$ by

$$\lambda_i = \frac{1}{n_i} \sum_{j=0}^{n_i - 1} |x_i(j)|, \quad 1 \leq i < L' \tag{3.78}$$

Under CASQ with multiple contexts, the models used are completely identical to the ones described above, but with careful consideration that there may be multiple DC frequencies whose corresponding sources $X_i$ should be modelled as uniform distributions. More formally, define $S_{DC} \subseteq \{0, 1, ..., L' - 1\}$ as the index set of all $i$ corresponding to DC frequencies, $0 \leq i < L'$. Define $S_{AC} \subseteq \{0, 1, ..., L' - 1\}$ as the index set of all $i$ corresponding to AC frequencies. Then each $X_i$ where $i \in S_{DC}$ should be modelled as a uniform distribution, and each $X_i$ where $i \in S_{AC}$ should be modelled as a Laplacian distribution with parameter $\lambda_i$.

To compute $Q^* = \{q_0^*, q_1^*, ..., q_{(L'-1)}^*\}$, we first choose a maximum integer step size $q_{max}$. If the water level $d$ is greater than the source variance $\sigma_i^2$ for some $0 \leq i < L'$, then all coefficients at the $i$th frequency position are to be quantized to zero (since the theoretical rate contribution from coding $X_i$ is zero). While it may seem like a violation of the HDQ principle to force quantization of some coefficients to zero, note that in this case we are quantizing *every* coefficient at the $i$th frequency position to zero, which would in fact be equivalent to setting $q_i^*$ to a very large number (large enough that each coefficient in $X_i$ is implicitly quantized to zero); thus, it can still be considered HDQ. In practice, we manually force the quantization of each coefficient in $X_i$ to zero and instead set $q_i^*$ to an arbitrary number since the reconstructed values will always be zero (it is set to $q_{max}$ in [7]).

If the water level $d$ is not greater than the source variance $\sigma_i^2$ for some $0 \leq i < L'$, then

we instead select $q_i^*$ according to the following:

$$q_i^* = \begin{cases} \min\left\{\left\lfloor\sqrt{12d}\right\rfloor, q_{max}\right\} & \text{if } i \in S_{DC} \\ \max\left\{q_i \in \mathcal{Q} : D_{Lap}(\lambda_i, q_i) \leq d\right\} & \text{if } i \in S_{AC} \end{cases} \tag{3.79}$$

where $\mathcal{Q} = \{1, 2, ..., q_{max}\}$. It can be shown that $D_{Lap}(\lambda_i, q_i)$ is a strictly increasing function of $q_i$. Therefore, the maximization above can easily be solved by a bisection search. This takes negligible time in practice due to the small search space and simple computation of $D_{Lap}$.

This algorithm is summarized below in Algorithm 1, which is almost identical to Algorithm 1 in [7].

**Algorithm 1** Optimal quantization table design for JPEG-type DCT-based coding under CASQ

Predetermine a desired distortion level $D_T$ and maximum quantization step size $q_{max}$
Predetermine the CASQ context selection for each block in the image
Determine $S_{DC}$ and $S_{AC}$ according to the above context selection
Determine the water level $d$ according to (3.50) and (3.51)
**for** each $i$, $0 \leq i < L'$ **do**
    **if** $\sigma_i^2 < d$ **then**
        set $q_i^* = q_{max}$
        signal that all coefficients in $X_i$ be quantized to zero
    **else**
        **if** $i \in S_{DC}$ **then**
            set $q_i^* = \min\{\lfloor\sqrt{12d}\rfloor, q_{max}\}$
        **else**
            set $q_i^* = \max\{q_i \in \mathcal{Q} : D_{Lap}(\lambda_i, q_i) \leq d\}$
        **end if**
    **end if**
**end for**

## 3.4 Context-Dependent SDQ

In this section, we propose a quantizer optimization method in the SDQ setting for CASQ when applied to JPEG-style DCT-based image coding. While the OptD theory in Sec-

tion 3.3 needed to be re-derived for HDQ CASQ based on [7], it turns out that the theoretical and practical aspects of our SDQ solution are nearly identical to [6] and [8]; only subtle modifications are necessary for their application to CASQ. In the following sub-sections, we separately discuss the Huffman and ARL cases.

Note that we still only consider the AC coefficients in the SDQ setting. Assume that there are $n_C$ quantization contexts. Since we are considering JPEG-style DCT-based coding in this section, assume that the 8×8 DCT is applied to each block. Assume that the AC frequencies associated with quantization context $c$, $0 \leq c < n_C$, are indexed by $i$, $1 \leq i < 64$, in zig-zag order.

### 3.4.1 Huffman Case

Recall the iterative SDQ solution discussed in Section 2.3.2, which comes from [6]:

Step 1) Determine an initial quantization table $Q^{(0)}$ from the given image and the corresponding run-size distribution $P_0$. Set $t = 0$ and specify a tolerance $\epsilon$ as the convergence criterion. Fix a value of $\lambda$.

Step 2) Fix $P^{(t)}$ and $Q^{(t)}$ for any $t \geq 0$. Find an optimal sequence $(r^{(t)}, s^{(t)}, id^{(t)})$ that achieves the following minimum:

$$\min_{(r,s,id)} \{J(\lambda) = D[(r, s, id)_{Q^{(t)}}] + \lambda R[(r, s), P^{(t)}]\} \tag{3.80}$$

Denote $J^{(t)}(\lambda)$ by $D[(r^{(t)}, s^{(t)}, id^{(t)})_{Q^{(t)}}] + \lambda R[(r^{(t)}, s^{(t)}), P^{(t)}]$. For $t > 0$, if $J^{(t-1)}(\lambda) - J^{(t)}(\lambda) \leq \epsilon$, stop the iterative algorithm and output $(r^{(t)}, s^{(t)}, id^{(t)})$ and $Q^{(t)}$. Otherwise, continue to Step 3.

Step 3) Fix $(r^{(t)}, s^{(t)}, id^{(t)})$. Update $Q^{(t)}$ and $P^{(t)}$ into $Q^{(t+1)}$ and $P^{(t+1)}$ respectively so that $Q^{(t+1)}$ and $P^{(t+1)}$ together achieve the following minimum:

$$\min_{Q,P} \{J(\lambda) = D[(r^{(t)}, s^{(t)}, id^{(t)})_Q] + \lambda R[(r^{(t)}, s^{(t)}), P]\} \tag{3.81}$$

where the above minimization is taken over all quantization tables $Q$ and all run-size probability distributions $P$. Note that $P^{(t+1)}$ can be selected as the empirical run-size distribution of $(r^{(t)}, s^{(t)})$.

Step 4) Return to Step 2 with $t = t + 1$.

Under CASQ, we define one quantization table $Q_c$ per context, $0 \leq c < n_C$. Since there is one Huffman table for each context, we also define one run-size probability distribution $P_c$ for each context.

To solve the minimization in Step 2, the authors of [6] proposed an efficient graph-based algorithm, the details of which remain identical under CASQ and need not be reproduced here. However, its usage differs in a subtle way. The algorithm used in Step 2 can be run independently in a block-by-block manner, since the Lagrangian cost $J(\lambda)$ is block-wise additive given $Q$ and $P$. Under CASQ, each block being optimized has an associated quantization context; accordingly, care needs to be taken in passing the correct quantization table $Q_c^{(t)}$ and run-size distribution $P_c^{(t)}$, where $c$ is the context index for the current block. The algorithm itself remains identical once $Q_c^{(t)}$ and $P_c^{(t)}$ are given.

The minimization in Step 3 is easily solved in [6] in closed-form if the squared-error distortion measure is used. The solution remains identical to that in [6] under CASQ, but with some differences in notation which we describe here. Denote the "new" quantization table for context $c$ by $\hat{Q}_c$ (that is, we will update $Q_c$ into $\hat{Q}_c$). Denote the $i$th step size of $Q_c$ by $q_{c,i}$. Denote the number of blocks in the image associated with the $c$th context as $N_{B_c}$. Denote the DCT coefficient of the image at frequency position $i$ of the $j$th block associated with the $c$th context as $C_{c,i,j}$. Denote the corresponding quantized index of $C_{c,i,j}$ by $K_{c,i,j}$, such that the reconstructed value is $q_{c,i} \cdot K_{c,i,j}$.

The minimum in Step 3 is achieved when [6]

$$\hat{q}_{c,i} = \frac{\sum_{j=0}^{N_{B_c}-1} (C_{c,i,j} \cdot K_{c,i,j})}{\sum_{j=0}^{N_{B_c}-1} K_{c,i,j}^2}, \quad 1 \le i < 64, \quad 0 \le c < n_C \tag{3.82}$$

In consideration of the above discussion, we now summarize our SDQ solution under CASQ in the following steps, which are essentially the same as the steps described earlier but with correct syntax under CASQ.

Step 1) Determine initial quantization tables $Q_c^{(0)}$ from the given image and the corresponding run-size distributions $P_c^{(0)}$, $0 \le c < n_C$. Set $t = 0$ and specify a tolerance $\epsilon$ as the convergence criterion. Fix a value of $\lambda$.

Step 2) Fix $P_c^{(t)}$ and $Q_c^{(t)}$ for every $0 \le c < n_C$ and for any $t \ge 0$. Find an optimal sequence $(r^{(t)}, s^{(t)}, id^{(t)})$ that achieves the following minimum:

$$\min_{(r,s,id)} \left\{ J(\lambda) = D\left[(r,s,id)_{\left\{Q_c^{(t)}\right\}_{c=0}^{n_C-1}}\right] + \lambda R\left[(r,s), \left\{P_c^{(t)}\right\}_{c=0}^{n_C-1}\right] \right\} \tag{3.83}$$

Denote $J^{(t)}(\lambda)$ by

$$J^{(t)}(\lambda) = D\left[(r^{(t)}, s^{(t)}, id^{(t)})_{\left\{Q_c^{(t)}\right\}_{c=0}^{n_C-1}}\right] + \lambda R\left[(r^{(t)}, s^{(t)}), \left\{P_c^{(t)}\right\}_{c=0}^{n_C-1}\right]$$

For $t > 0$, if $J^{(t-1)}(\lambda) - J^{(t)}(\lambda) \leq \epsilon$, stop the iterative algorithm and output $(r^{(t)}, s^{(t)}, id^{(t)})$ and $Q_c^{(t)}$, $0 \leq c < n_C$. Otherwise, continue to Step 3.

Step 3) Fix $(r^{(t)}, s^{(t)}, id^{(t)})$. Update $Q_c^{(t)}$ and $P_c^{(t)}$ into $Q_c^{(t+1)}$ and $P_c^{(t+1)}$ respectively so that $Q_c^{(t+1)}$ and $P_c^{(t+1)}$ together achieve the following minimum:

$$\min_{\{Q_c\}_{c=0}^{n_C-1}, \{P_c\}_{c=0}^{n_C-1}} \Big\{ J(\lambda) = D\left[(r^{(t)}, s^{(t)}, id^{(t)})_{\{Q_c\}_{c=0}^{n_C-1}}\right] + \lambda R\left[(r^{(t)}, s^{(t)}), \{P_c\}_{c=0}^{n_C-1}\right] \Big\} \tag{3.84}$$

where the above minimization is taken over all quantization tables $Q_c$ and all run-size probability distributions $P_c$, $0 \leq c < n_C$. Note that $P_c^{(t+1)}$ can be selected as the empirical run-size distribution of $(r^{(t)}, s^{(t)})$.

Step 4) Return to Step 2 with $t = t + 1$.

**Initial Quantization Table Selection**

Recall from [6] that the iterative solution only converges to a local minimum. The choice of initial quantization tables has a significant impact on the overall RD performance under SDQ. The optimal choice of initial quantization tables is an open problem. We propose using the table $Q$ generated by our modified OptD method in Section 3.3 as the intial table, where the target distortion $D_T$ is chosen such that the bit-rate resulting from the HDQ of the image using $Q$ is slightly higher than the target bit-rate corresponding to $\lambda$ under SDQ. The purpose of targeting a slightly higher bit rate with OptD is to "give room" for subsequent optimization under SDQ. Our experiments have shown that this is a reasonably good choice of initial quantization table.

### 3.4.2 ARL Case

Likewise in the ARL case, our solution is nearly identical to the one proposed in [8], with only some minor changes to the syntax.

Denote $P(M_c)$ as the collection of context model distributions for $M_c$. We summarize our solution in the steps below:

Step 1) Determine initial quantization tables $Q_c^{(0)}$ from the given image and the corresponding context model distributions $P(M_c^{(0)})$. Set $t = 0$ and specify a tolerance $\epsilon$ as the convergence criterion. Fix a value of $\lambda$.

Step 2) Fix $P(M_c^{(t)})$ and $Q_c^{(t)}$ for every $0 \leq c < n_C$ and any $t \geq 0$. Find an optimal sequence $(r^{(t)}, l^{(t)})$ that achieved the following minimum:

$$\min_{(r,l)} \left\{ J(\lambda) = D\left[(r,l)_{\left\{Q_c^{(t)}\right\}_{c=0}^{n_C-1}}\right] + \lambda R((r,l), \{P(M_c^{(t)})\}_{c=0}^{n_C-1}) \right\} \tag{3.85}$$

Denote $J^{(t)}(\lambda)$ by

$$J^{(t)}(\lambda) = D\left[(r^{(t)}, l^{(t)})_{\left\{Q_c^{(t)}\right\}_{c=0}^{n_C-1}}\right] + \lambda R((r^{(t)}, l^{(t)}), \{P(M_c^{(t)})\}_{c=0}^{n_C-1}) \tag{3.86}$$

For $t > 0$, if $J^{(t-1)}(\lambda) - J^{(t)}(\lambda) \leq \epsilon$, stop the iterative algorithm and output $(r^{(t)}, l^{(t)})$ and $Q_c^{(t)}$, $0 \leq c < n_C$. Otherwise, continue to Step 3.

Step 3) Fix $(r^{(t)}, l^{(t)})$. Update $P(M_c^{(t)})$ and $Q_c^{(t)}$ into $P(M_c^{(t+1)})$ and $Q_c^{(t+1)}$ respectively so that $P(M_c^{(t+1)})$ and $Q_c^{(t+1)}$ together achieve the following minimum:

$$\min_{\{P(M_c)\}_{c=0}^{n_C-1}, \{Q_c\}_{c=0}^{n_C-1}} \Big\{ J(\lambda) = D\left[(r^{(t)}, l^{(t)})_{\{Q_c\}_{c=0}^{n_C-1}}\right] + \lambda R((r^{(t)}, l^{(t)}), \{P(M_c)\}_{c=0}^{n_C-1}) \Big\} \tag{3.87}$$

Step 4) Return to Step 2 with $t = t + 1$.

The minimization in Step 2 is achieved using the exact same block-wise graph-based dynamic programming algorithm from [8], where we only need to take care in passing the correct $P(M_c)$ and $Q_c$ to the algorithm corresponding to the context index $c$ of the current block. Updating $Q_c$, $0 \leq c < n_C$, in Step 3 is identical to the Huffman case as described in the previous section. $M_c$, $0 \leq c < n_C$, is updated based on the default arithmetic context selection method to be discussed in 4.4.2.

However, note that in [8] and as well in our solution, we use a fixed arithmetic context for the first bit of the first run, rather than the neighbour-dependent arithmetic contexts described in Section 2.1.3 and Section 4.4.2. This is because a full optimization over neighbour-dependent arithmetic contexts would greatly increase the complexity of the graph-based algorithm. Since coding the first bit of the first run contributes to a very small fraction of the bit rate, this change is justified.

Like in the Huffman case, initial quantization table selection is an issue here as well. We simply propose the same method as described previously: use the HDQ CASQ algorithm to generate initial quantization tables, where the bit rate resulting from HDQ is slightly higher than the target bit rate for SDQ.

## 3.5 Summary

In this chapter, we reviewed the shortcomings of JPEG regarding local adaptivity and proposed a new type of quantization framework, called context adaptive space quantization (CASQ), to address these shortcomings by quantizing the blocks of an image conditioned on a quantization context. We subsequently formulated and solved optimal quantizer design under CASQ in both the HDQ and SDQ settings. The quantizer optimization solutions presented in this chapter facilitate the development of a practical image coder based on CASQ, to be discussed in the next chapter.

# Chapter 4

# CASQ-Based Image Coder

In this chapter, we describe a new image coding scheme based on JPEG that incorporates the CASQ framework derived in Chapter 3.

## 4.1 Overview

Our image coder is summarized in Fig. 1.3. Similar to JPEG, our coder begins by partitioning an input image $I_0$, consisting of $N_{row}$ rows and $N_{col}$ columns of pixels, into non-overlapping 8×8 blocks. This is followed by a block-wise 8×8 DCT. The next logical step would be quantization under CASQ; but before that can occur, we first need to perform quantization context selection.

In general, context selection may be performed at any point before CASQ; it need not even be image-dependent. Any number of contexts $n_C$ may also be used. In our coding scheme we define two contexts, namely "homogeneous" (relatively flat and featureless) and "non-homogeneous" (relatively detailed features). We construct an online classifier based on the statistics of the DCT coefficients of $I_0$ for the purpose of assigning a context label to each block. The details relating to context selection will be discussed in Section 4.2. We may also alternatively refer to these contexts as "regions", where the "homogeneous region" refers to the set of blocks corresponding to the homogeneous context and the "non-homogeneous region" refers to the set of blocks corresponding to the non-homogeneous context.

After context selection has been completed by the classifier, we use the CASQ method proposed in Chapter 3 for quantizing the DCT coefficients. Finally, the quantized indices

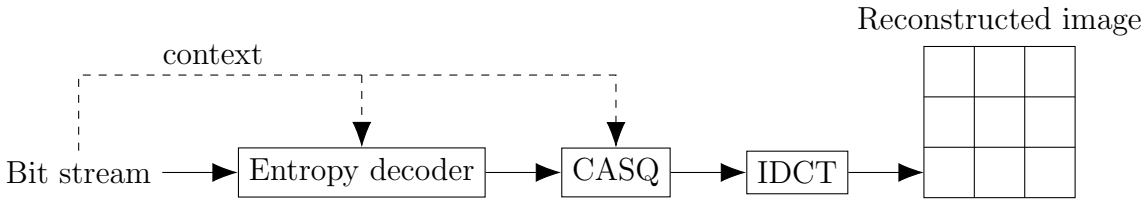



Figure 4.1: Proposed image coding scheme (decoder)

from each block, along with their associated context labels, are passed to the entropy coder (to be discussed in Section 4.4) in raster order to form the final bit stream. We propose both JPEG-style Huffman coding and ARL coding for this purpose. The quantization tables, Huffman tables (in the Huffman case) and context labels are transmitted as side information. Encoding of the quantization tables and Huffman tables is identical to JPEG. Encoding of the context labels will be discussed in detail in Section 4.2.2.

The decoding process (summarized in Fig. 4.1) is essentially the reverse of the encoding process, except that classification is not performed since the context labels are available to the decoder as side information. First, each compressed block is passed to an entropy decoder (either Huffman or ARL) along with its context label, resulting in the quantized indices for that block. Then, the quantized indices for that block are dequantized according to the uniform reconstruction rule in JPEG using the quantization table $Q_c$, where $c$ is the context index associated with that block. Finally, once all blocks have been dequantized, the inverse DCT is performed and the reconstructed image is sent to the output.

## 4.2 Context Selection

Context selection is the process whereby a context index $c$, $0 \leq c < n_C$, is selected for each block. In our image coding scheme, $n_C = 2$. We only loosely describe these contexts as "relatively flat and featureless" and "relatively detailed", respectively. The idea behind this distinction is that flatter blocks will tend to have lower energy in the AC frequencies, and thus would benefit more in the RD sense from a quantization table with higher step sizes. On the other hand, more detailed blocks will tend to have higher energy in the AC frequencies, which would benefit more from a quantization table with lower step sizes.

Motivated by this idea, we seek a context selection method that can distinguish blocks between "flat" and "detailed". To this end, we apply Laplacian transparent composite

modelling (LPTCM) to detect blocks which contribute significantly to the perceptual details of the image. The details of LPTCM and its application to context selection will be discussed in the following sub-sections.

### 4.2.1 Laplacian Transparent Composite Modelling

Laplacian transparent composite modelling (LPTCM) is a type of statistical model which can accurately describe the distribution of DCT coefficients [13]. Similar to [14], we define one model for each of the 63 AC frequency positions. We ignore the DC frequency because it does not capture any information about intra-block details.

We define a zero-mean probability density function $f_k(y)$, $1 \leq k < 64$, corresponding to the $k$th frequency position (in zig-zag order). The purpose of $f_k(y)$ is to model the statistics of all of the DCT coefficients at the $k$th frequency. LPTCM defines two parts for this density: a central truncated Laplacian part and an outer, uniformly distributed tail part. The density function is given by

$$f_k(y) = \begin{cases} \frac{b_k}{2\lambda_k\left(1-e^{(-Y_k/\lambda_k)}\right)} e^{-\frac{|y|}{\lambda_k}}, & \text{if } |y| < Y_k \\ \frac{1-b_k}{2(A_k-y_k)}, & \text{if } Y_k < |y| \leq A_k \\ 0, & \text{if } |y| > A_k \end{cases} \tag{4.1}$$

where $A_k$ denotes the maximum magnitude among the DCT coefficients at frequency $k$, $0 \leq b_k \leq 1$ is a constant denoting the fraction of coefficients which fall into the Laplacian part, $\lambda_k$ is the scale parameter for the Laplacian part, and $Y_k$ is the cut-off boundary between the Laplacian part and the uniform part. Coefficients which fall into the central part are referred to as "inliers", and those which fall into the tail part are referred to as "outliers". A schematic plot of $f_k(y)$ is shown in Fig. 4.2, where the scaling has been exaggerated in order to more easily display the important parameters of $f_k(y)$.

The parameter $A_k$ can be determined by a simple scan of the DCT coefficients:

$$A_k = \max |C_{k,l}|, \quad 1 \leq k < 64 \tag{4.2}$$

where $C_{k,l}$ denotes the DCT coefficient at frequency position $k$ in the $l$th block. The remaining parameters $b_k$, $\lambda_k$ and $Y_k$ can be efficiently estimated online using Algorithm 1 and Algorithm 2 in [13]. Note that since each frequency position is modelled independently, then parameter estimation over all frequencies can easily be implemented in a parallel manner.

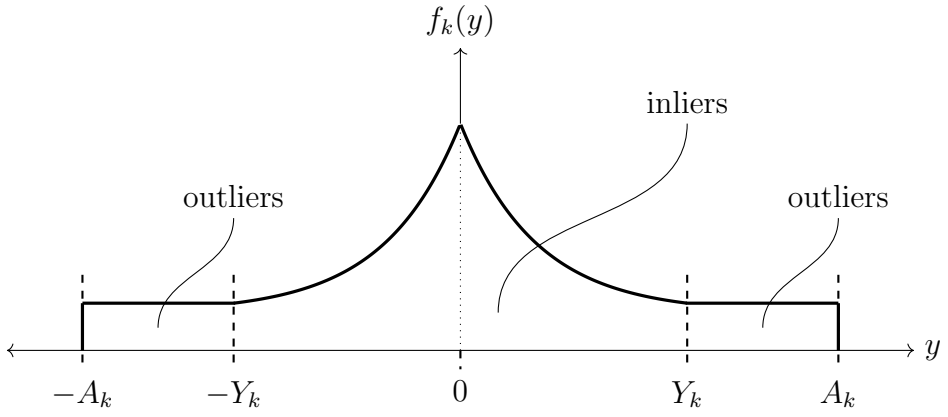



Figure 4.2: Schematic plot of $f_k(y)$

### 4.2.2 Classification and Outlier Map

Our classifier is built upon the separation between inlier coefficients and outlier coefficients as defined by LPTCM. Assume that the LPTCM parameters for each of the AC frequencies are given (for example, by the estimation algorithms in [13]). We define a classifier $L(B) : \mathbb{R}^{64} \longrightarrow \{\text{“homogeneous”, “non-homogeneous”}\}$, where $B$ is an 8×8 block of DCT coefficients, as follows. Denote the 8×8 block of DCT coefficients at the $i$th block-row and $j$th block-column by $B_{i,j}$, $0 \leq i < N_{row}/8$, $0 \leq j < N_{col}/8$. Denote the coefficient at the $k$th frequency position (in zig-zag order) in block $B_{i,j}$ by $(B_{i,j})_k$. Define $T(B_{i,j})$ as the number of outlier coefficients contained within $B_{i,j}$:

$$T(B_{i,j}) = \sum_{i=1}^{63} I_k((B_{i,j})_k) \tag{4.3}$$

where $I_k(y)$ is an indicator function:

$$I_k(y) = \begin{cases} 0, & |y| < Y_k \\ 1, & |y| \geq Y_k \end{cases} \tag{4.4}$$

Our classifier is then defined by

$$L(B_{i,j}) \triangleq \begin{cases} \text{“homogeneous”}, & \text{if } T(B_{i,j}) = 0 \\ \text{“non-homogeneous”}, & \text{if } T(B_{i,j}) > 0 \end{cases} \tag{4.5}$$

In other words, if at least one AC coefficient in a given block falls into the outlier region defined by its respective LPTCM model, then we classify the block as non-homogeneous. Otherwise, all AC coefficients in the block fall into their respective inlier regions and we classify the block as homogeneous.

Context selection, therefore, is the result of applying this classifier to every block in the image. The output of the classifier for each block is a context label which must be provided to both CASQ and the entropy coder. To help us draw a connection to CASQ as described in Chapter 3, we can define a mapping between $L \in \{\text{“homogeneous”}, \text{“non-homogeneous”}\}$ and $c \in \{0, 1\}$, where $L$ is a context label and $c$ is its corresponding context index.

### Outlier Map

The output of the classifier must also be transmitted to the decoder as side information. To this end, we define an “outlier map” denoted by $m(i, j)$ to be the binarized representation of the result of applying the classifier $L$ to the block at the $i$th row and $j$th column:

$$m(i,j) \triangleq \begin{cases} 0, & \text{if } L(B_{i,j}) = \text{“homogeneous”} \\ 1, & \text{if } L(B_{i,j}) = \text{“non-homogeneous”} \end{cases} \tag{4.6}$$

We can think of $m(i, j)$ as a two-level image consisting of $N_{row}/8$ rows and $N_{col}/8$ columns of pixels. Note that $m(i, j)$ is only 1/64th the area of $I_0$ and contains only binary values; as such, it tends to contribute very little to the overall bit rate when compared to compressing $I_0$. For simplicity, we use an adaptive binary arithmetic coder with 16 arithmetic coding contexts to compress $m(i, j)$. We code $m(i, j)$ for every block-row $i$ and every block-column $j$ in $I_0$ in raster order, $0 \leq i < N_{row}/8$, $0 \leq j < N_{col}/8$. The arithmetic coding context used to code $m(i, j)$ depends on its four causal neighbours:

$$\begin{aligned} \text{coding-context}(i,j) = {} & 1 \cdot m(i-1, j-1) \\ & + 2 \cdot m(i-1, j) \\ & + 4 \cdot m(i-1, j+1) \\ & + 8 \cdot m(i, j-1) \end{aligned} \tag{4.7}$$

Since each $m(i, j)$ has a binary state then there are 16 possible state combinations of the four causal neighbours, hence there are 16 arithmetic coding contexts.

An example outlier map generated from our classifier when applied to the Lena image is shown in Fig. 4.3(b), with the original image shown for reference in Fig. 4.3(a). A

value of "0" from the outlier map is represented by a black pixel, while a value of "1" is represented by a white pixel. We observe that the more detailed parts of the image, such as sharp edges, facial features and the part of the hat hanging down, have been classified as non-homogeneous; while the more flat areas, including most of the background, have been classified as homogeneous. Around 26% of the blocks have been classified as non-homogeneous in this example.

(a) Lena, original image (b) Lena, outlier map

Figure 4.3: Lena with corresponding outlier map

## 4.3 Quantization

Given the output of running the classifier on each block in $I_0$, we can proceed with quantization under CASQ. In our image coder, either HDQ or SDQ may be considered for quantizer optimization; optimal quantizer design in both settings under CASQ was described in detail in Chapter 3.

If HDQ is used, then we apply Algorithm 1 described in Section 3.3, where we set $n_C = 2$ ($L' = 128$). The output of this process is a set of quantization tables $\{Q_0, Q_1\}$, where $Q_0$ corresponds to the homogeneous region and $Q_1$ corresponds to the non-homogeneous region. We then quantize all blocks of DCT coefficients in the homogeneous region using $Q_0$ and the HDQ mapping from (2.6). Similarly, we quantize all blocks in the non-homogeneous region using $Q_1$. These quantized blocks are then entropy coded using either Huffman or ARL coding as described in the next section.

If SDQ is used, then we apply the optimization solution from Section 3.4 (specifically, we apply the solution corresponding to either Huffman or ARL coding, depending on which entropy coding method is being used by our image coder), again with $n_C$ set to 2. This process will produce not only the quantization tables $Q_0$ and $Q_1$, but it will also produce all of the quantized DCT coefficients across all blocks and all quantization contexts. Since quantization has already been completed, we can proceed directly to entropy coding.

## 4.4 Entropy Coding

Entropy coding is the process whereby quantized DCT coefficients are losslessly compressed to form the final bit stream. In our image coder, we employ two different kinds of entropy coding, namely Huffman coding and adaptive runlength (ARL) coding. In both the Huffman and ARL cases, compression is achieved by exploiting the empirical statistics of the quantized DCT coefficients, which depend directly on the quantization method used. Given that blocks may be quantized differently from one another under CASQ depending on their respective quantization contexts, then it makes sense to perform entropy coding conditioned on those quantization contexts as well.

In this section, we describe entropy coding for our image coder under CASQ. We base our methods on those already been described in detail in Section 2.1; since the core details of these methods have already been discussed, we will focus only on how they are different under CASQ.

### 4.4.1 Context-Dependent Huffman Coding

Huffman coding for JPEG was already described in Section 2.1.2. To apply JPEG-style Huffman coding to our image coder under CASQ, we define separate AC Huffman tables $H_0$ and $H_1$ corresponding to the homogeneous and non-homogeneous regions, respectively; note that this was alluded to in our discussion of CASQ in the SDQ setting in Section 3.2.2. Since there is a separate step size defined for the DC frequency position for each context, we also define separate DC Huffman tables $H_c^{DC}$. All Huffman tables for all contexts are transmitted to the decoder as side information in the same manner as JPEG.

In our image coder, all blocks corresponding to the $c$th quantization context are compressed using the exact same run-length and Huffman coding method from JPEG, where the DC Huffman table $H_c^{DC}$ and the AC Huffman table $H_c$ are used. Other than selecting which tables to use based on $c$, the encoding process is identical to that of JPEG.

To actually determine $H_c$ and $H_c^{DC}$ for the two quantization contexts, we use the customized method from [2] whereby the tables are optimized based on the statistics of the quantized DCT coefficients. To build $H_c$, the statistics from the quantized AC coefficients corresponding to context $c$ only are used. Similarly, to build $H_c^{DC}$ the statistics from the quantized DC coefficients corresponding to context $c$ only are used. This is done for each of the two quantization contexts.

### 4.4.2 Context-Dependent ARL Coding

ARL coding for JPEG was already described in Section 2.1.3. In ARL coding, the empirical statistics of the quantized DCT coefficients (or rather, of the run-level values generated from the quantized coefficients) are gathered by adaptively updating the probability distribution models contained inside the arithmetic coder. These updates are performed once for every bit coming into the encoder. The coder keeps track of a separate model for each of the 32 arithmetic coding contexts defined by ARL. Hence, if we wish to exploit the differing statistics of quantized coefficients between different quantization contexts, then we must define separate arithmetic coding contexts as well.

To this end, we start with the most obvious choice: simply duplicate the set of 32 arithmetic coding context models used by ARL, so that there is one set of 32 models corresponding to the homogeneous region and another set of 32 models corresponding to the non-homogeneous region. Then, if we are encoding a run-level value coming from a block that corresponds to the homogeneous region, we pick the arithmetic coding context model from the set of 32 models associated with the homogeneous region. Similarly, if we are encoding a value from the non-homogeneous region, then we pick the model from the set of 32 models associated with the non-homogeneous region. Within each of these sets, we would use the same process as ARL described in Section 2.1.3 to determine which of the 32 arithmetic coding contexts to use.

However, recall that some of the models used in ARL depend on properties of neighbouring blocks. Specifically, to code whether or not each DC residue value is zero, one of three models is used depending on whether the upper and/or the left neighbouring blocks also have zero residue. Since these blocks may be associated with different quantization contexts from the current block, then these three models may not be sufficient. For example, if the current block is from the homogeneous region and one of the neighbouring blocks is from the non-homogeneous region, then it is more likely that the DC residue value for the current block will be non-zero since the DC coefficients may have been quantized using different step sizes.

We therefore propose a further modification to the context models defined under ARL. Denote $a$ as a binary flag indicating whether or not either of the upper or left neighbouring blocks is associated with a quantization context that is different from the current block ($a = 0$ if both neighbouring blocks have the same quantization context as the current block and $a = 1$ otherwise). Recall from Section 2.1.3 that $z$ is defined as the number of neighbouring blocks which contain non-zero quantized DC residue values. Then, we modify the aforementioned DC context models as follows:

$$\begin{aligned}&(z = 0 \text{ and } a = 0)(z = 1 \text{ and } a = 0)(z = 2 \text{ and } a = 0)\\&(z = 0 \text{ and } a = 1)(z = 1 \text{ and } a = 1)(z = 2 \text{ and } a = 1)\end{aligned} \tag{4.8}$$

This brings the total number of context models from 32 to 35 for each of the two quantization contexts.

Similarly, we also propose a modification to the three context models used to code the first bit of the first run value. Recall that $f$ is defined as the number of neighbouring blocks which contain non-zero quantized AC coefficients. We modify those three models as follows:

$$\begin{aligned}&(f = 0 \text{ and } a = 0)(f = 1 \text{ and } a = 0)(f = 2 \text{ and } a = 0)\\&(f = 0 \text{ and } a = 1)(f = 1 \text{ and } a = 1)(f = 2 \text{ and } a = 1)\end{aligned} \tag{4.9}$$

Thus, the total number of context models for each of the two quantization contexts is actually 38. Since there are two quantization contexts, then the grand total number of context models over all quantization contexts is 76.

### 4.4.3 JPEG Compatibility

It should be noted that our image coder in Huffman mode is not directly compatible with a baseline JPEG coder. However, it may be implemented instead as a simple front-end to any existing baseline JPEG coder. The encoding process of such a front-end may be as follows:

1. Perform the DCT and context selection
2. Apply CASQ to obtain the optimal $Q_0$ and $Q_1$
3. Assemble the blocks from the homogeneous region into an image consisting of a single row (or column) of blocks, in some predefined order
4. Use any baseline JPEG encoder to compress this single-row (or single-column) homogeneous image using $Q_0$ and a customized Huffman table

5. Transmit the resulting bit stream, which is JPEG-compatible
6. Repeat steps 3 to 5 for the non-homogeneous region
7. Compress and transmit the outlier map

A decoder front-end would receive two JPEG-compatible bit streams (one for each region) in addition to the compressed outlier map. Any baseline JPEG decoder can decompress the two image streams, which may then be reassembled into the original image using the outlier map.

## 4.5 Experimental Results

### 4.5.1 RD Performance

We tested our image coder over the following input parameters:

- Input image
- Target bit rate
- Quantization mode (HDQ or SDQ)
- Entropy mode (Huffman or ARL)

For the input image, we tested seven standard test images: 512×512 Lena, 512×512 Airplane (F16), 512×512 Goldhill, 2048×2560 Bike, 2048×2560 Woman, 1280×720 Stockholm (first frame) and 1920×1080 Kimono (first frame). For the bit rates, we tested eight different rates spaced uniformly from 0.25 bpp (bits per pixel) to 2 bpp. Both of the HDQ and SDQ modes were tested. Both Huffman coding and ARL coding were tested.

For each input image, target bit rate, quantization mode and entropy mode, the following test was performed. The input image was compressed to the target bit rate using our image coder, where either HDQ or SDQ was used for quantization and either Huffman or ARL coding was used for entropy coding. The configurations corresponding to the quantization mode and the entropy mode are abbreviated as "Huffman-HDQ-CASQ", "Huffman-SDQ-CASQ", "ARL-HDQ-CASQ" and "ARL-SDQ-CASQ". The compressed image was then reconstructed and the distortion between the reconstructed image and the original image was measured. We used the peak signal-to-noise ratio (PSNR) distortion measure, in units of dB (decibels). For all tests, the overhead incurred from transmitting quantization tables, Huffman tables and the outlier map are included in the reported bit rate.

In order to judge the RD performance of our image coder, we compared our RD results from each of the aforementioned tests to the RD results from performing the exact same

tests on some benchmark image coders instead. We selected three benchmarks in the HDQ setting and one benchmark in the SDQ setting. For HDQ, the first benchmark was baseline JPEG coding where the sample quantization table from [2] was scaled by some constant to achieve rate control. The configurations for this benchmark are abbreviated as "Huffman-HDQ-JPEG" and "ARL-HDQ-JPEG". The second benchmark, representing the current state-of-the-art HDQ method, was baseline JPEG coding using the OptD method from [7] for quantization table selection. The configurations for this benchmark are abbreviated as "Huffman-HDQ-OptD" and "ARL-HDQ-OptD". The third benchmark was JPEG2000 using the JasPer [15] codec with default settings (6 levels of DWT). In the Huffman case, the comparisons between our image coder and the three HDQ benchmarks are abbreviated as "H1", "H2" and "H3", respectively. In the ARL case, these comparisons are abbreviated as "A1", "A2" and "A3". For SDQ, the benchmark was the baseline JPEG SDQ method from [6] in the Huffman case and from [8] in the ARL case. The configurations for this benchmark are abbreviated as "Huffman-SDQ" and "ARL-SDQ". In the Huffman case, the comparison between our image coder and the SDQ benchmark is abbreviated as "H4". In the ARL case, this comparison is abbreviated as "A4".

The average PSNR gains resulting from comparing our image coder to the benchmarks are summarized in Table 4.1. An average positive gain is reported for nearly every image under test against every benchmark, except for JPEG2000 in the Huffman case. In the cases of Goldhill H4 and Kimono H4, a very slight loss is reported. These two particular images are much more detailed from the other ones and do not benefit as much from CASQ, although CASQ still performs at least as well as the benchmarks regardless; the observed losses can be attributed to the overhead incurred in transmitting the outlier map.

The average overall gains are very promising: almost 3 dB PSNR gain is observed in the case of Woman using ARL coding when compared to baseline JPEG with no quantizer optimization. The average gain across all images tested in this configuration is also very significant at 2.00 dB. When ARL coding is used, we tend to see higher gains compared to when Huffman coding is used. Even for the more "difficult" images, CASQ still performs at least as well as the previous state-of-the-art quantizer optimization found in [6–8]. These gains still have the potential to be exploited even further if we consider that more advanced context selection techniques than the one proposed in this thesis are possible.

The results further show that our image coder is comparable to JPEG2000 in RD performance when Huffman coding is used, and it outperforms JPEG2000 by over 0.6 dB PSNR on average when ARL coding is used.

Detailed results for each test in the form of RD tables can be found in Section A.1 of Appendix A. Sample RD curves for Lena, Kimono and Woman (corresponding to the

Table 4.1: Average PSNR gain (in dB) summary

| | Huffman | | | ARL | | | JPEG2000 | |
|---|---|---|---|---|---|---|---|---|
| Image | H1 | H2 | H4 | A1 | A2 | A4 | H3 | A3 |
| Lena | 1.63 | 0.17 | 0.06 | 1.89 | 0.30 | 0.14 | 0.053 | 0.67 |
| Airplane | 1.94 | 0.35 | 0.14 | 2.24 | 0.44 | 0.25 | -0.039 | 0.58 |
| Goldhill | 1.50 | 0.02 | -0.008 | 1.65 | 0.05 | 0.004 | -0.10 | 0.48 |
| Bike | 2.34 | 0.47 | 0.23 | 2.58 | 0.54 | 0.32 | -0.52 | 0.00058 |
| Woman | 2.63 | 0.53 | 0.19 | 2.89 | 0.54 | 0.23 | -0.021 | 0.67 |
| Stockholm | 1.41 | 0.05 | 0.02 | 1.58 | 0.09 | 0.06 | 0.062 | 0.57 |
| Kimono | 1.03 | 0.003 | -0.009 | 1.19 | 0.04 | 0.03 | 0.79 | 1.31 |
| Average | 1.78 | 0.23 | 0.09 | 2.00 | 0.28 | 0.15 | 0.032 | 0.61 |

average, worst and best cases, respectively) coded using HDQ and Huffman modes are shown in Fig. 4.4, Fig. 4.5 and Fig. 4.6, respectively.

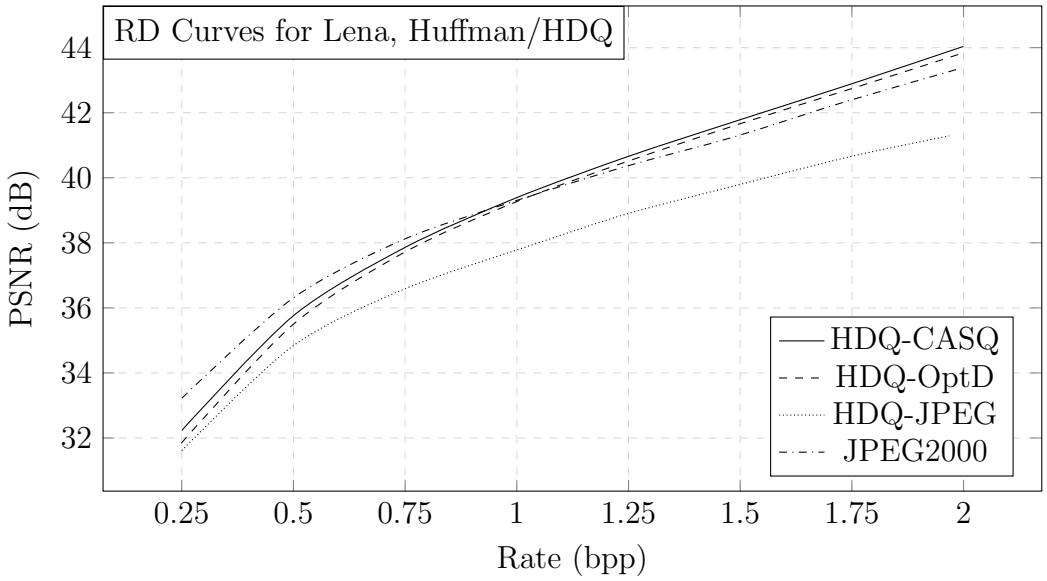



Figure 4.4: RD curves for Lena using Huffman coding and HDQ

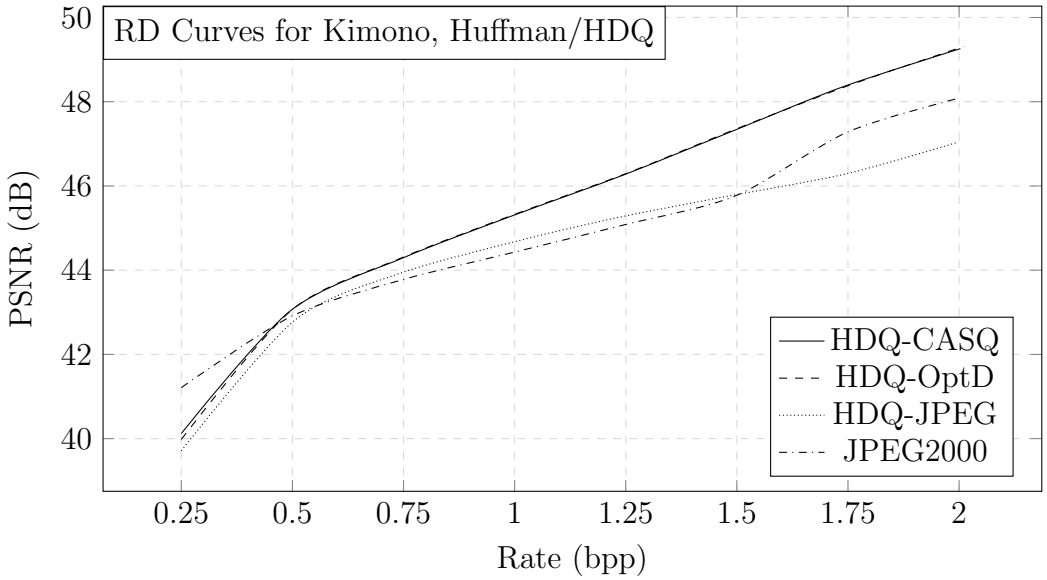



Figure 4.5: RD curves for Kimono using Huffman coding and HDQ

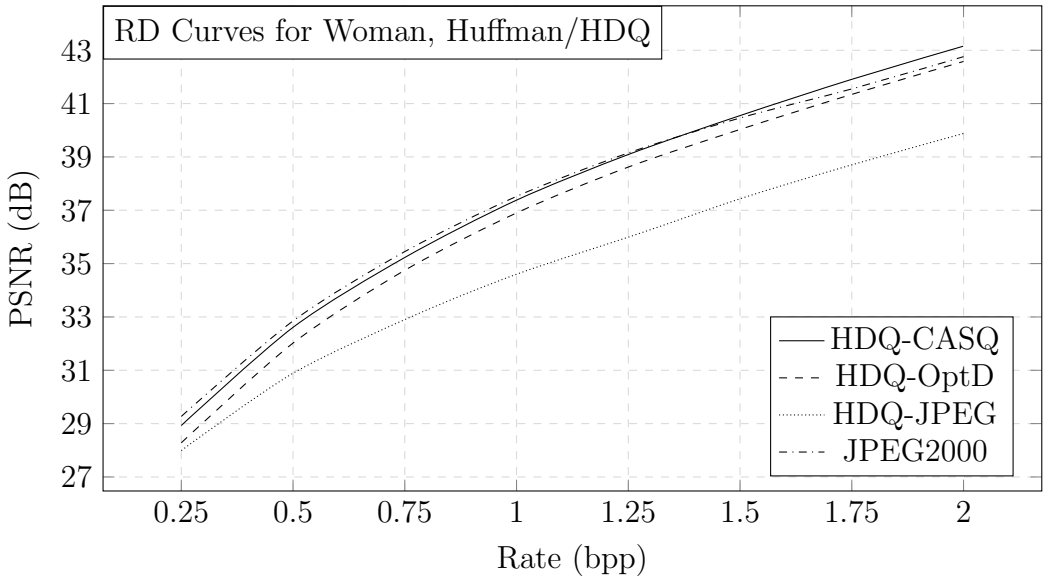



Figure 4.6: RD curves for Woman using Huffman coding and HDQ

To get an idea of the subjective performance of our image coder, a comparison is shown in Figs. 4.7(a)-(d). In Figs. 4.7(b)-(d), the Lena image was compressed to a bit rate of 0.25 bpp using different quantization methods and a portion of the resulting reconstructed image is shown. The original image is shown in Fig. 4.7(a) for reference. In Fig. 4.7(b), the standard JPEG table (scaled by a constant to achieve rate control) was used. We observe that many distortion artifacts are present, especially along the edges and in the eyes. In Fig. 4.7(c), the OptD method was used to optimize the quantization table. Although much of the artifact distortion has been mitigated, it is still difficult to discern some details, especially in and around the eyes. In Fig. 4.7(d), CASQ optimization was applied. The irises appear to be more defined and the eyelashes are much more clearly visible compared to the other versions. Parts of the hat also appear to be more clear.

(a) Original

(b) Standard JPEG table; PSNR = 31.61 dB

(c) OptD; PSNR = 31.84 dB

(d) CASQ; PSNR = 32.23 dB

Figure 4.7: Subjective comparison of Lena at 0.25 bpp between different quantization methods

### Outlier Map

All of the above numbers and figures include the bit rate overhead incurred from compressing the outlier map. To help get an idea of how significant the outlier map is in the RD

sense, we show the bit rates of the compressed outlier maps for each image in Table 4.2. Note that since the outlier map is compressed losslessly and depends only on context selection, it is not affected by the target bit rate of the image; hence, the relative contribution of the outlier map to the overall bit rate becomes smaller as the target bit rate increases. This is shown in Figure 4.8 for Lena.

Table 4.2: Compressed outlier map bit rates

| Image | Rate (bpp) |
|---|---|
| Lena | 0.00769 |
| Airplane | 0.00769 |
| Goldhill | 0.00763 |
| Bike | 0.00785 |
| Woman | 0.00482 |
| Stockholm | 0.00874 |
| Kimono | 0.00882 |
| Average | 0.00761 |

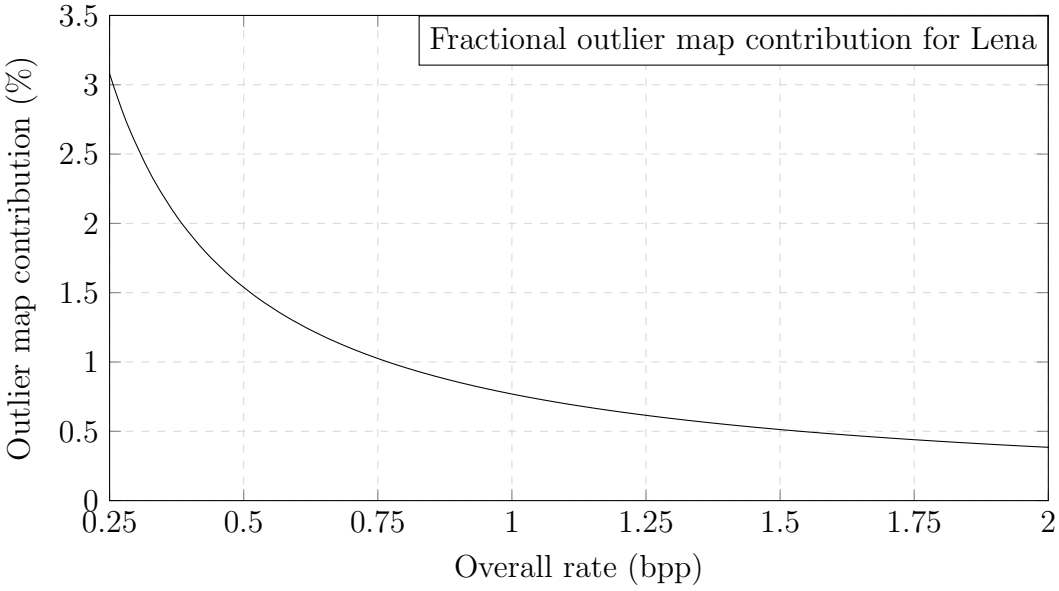



Figure 4.8: Relative outlier map rate contribution for Lena

### 4.5.2 Computational Performance

One of the largest contributors to additional complexity incurred by our image coder (when compared to baseline standard JPEG coding) is LPTCM parameter estimation, which is necessary for the online classifier. Quantizer optimization and compressing the outlier map also add some computational overhead to our image coder; overhead which is absent when using a baseline JPEG coder with the sample quantization table. These processes encapsulate nearly all of the additional computational complexity incurred by our image coder when compared to baseline JPEG coding without quantizer optimization. The DCT process is identical to JPEG and the complexity of our entropy coders has not changed either. We are only interested in the *additional* complexity incurred from our image coder: to this end, we focus on the computational complexity of the LPTCM estimation, quantizer optimization and compressing the outlier map. Note that the first two processes are performed only at the encoder side: the additional complexity is completely absent at the decoder side. The complexity from compressing (and decompressing at the decoder side) the outlier map is very low, as we will see shortly.

For LPTCM parameter estimation, we measured the time taken to estimate the LPTCM parameters across all 63 AC frequency positions using the algorithms presented in [13]. Estimation using these algorithms requires the input data to be sorted; we have included the sorting time in our measurements, as it in fact contributes a majority to the overall computation time.

For HDQ quantizer optimization, we measured the time taken to gather the DCT statistics and the time taken to compute the quantization table according to Algorithm 1. $D_T$ was arbitrarily chosen to be 10 and $q_{max}$ was set to 46. For SDQ quantizer optimization, we measured the time taken to perform the iterative optimization algorithm described in Section 3.4.1 in the case of Huffman coding and in Section 3.4.2 in the case of ARL coding. A convergence criterion value of $\epsilon = 0.1$ was used for SDQ. We do not include the time used for initial quantization table selection, which itself (in our image coder) amounts to HDQ optimization anyway.

For compressing the outlier map $m(i, j)$, we measured the time taken to compress $m(i, j)$, $0 \leq i < N_{row}/8$, $0 \leq j < N_{col}/8$, into a compressed bit stream as described in Section 4.2.2, as well as the time taken to subsequently decompress it.

For the sake of comparison between our HDQ quantizer optimization algorithm (CASQ with two contexts) and OptD from [7], we have also measured the time taken to run Algorithm 1 from [7] on the same input images. We found that the time difference is negligible: our quantizer optimization over two contexts (from Section 3.3) is just as fast

as the original OptD algorithm from [7]. Likewise, for comparison in the SDQ setting, we have also measured the time taken to perform the iterative optimization algorithm described in [6] in the Huffman case and [8] in the ARL case; our method runs in roughly the same time as the benchmarks.

All tests were performed on an Intel i7-4790 processor at 3.60 GHz, with multi-threading additionally enabled for LPTCM parameter estimation and SDQ block-wise run-size sequence optimization (step 2 of the iterative SDQ algorithms in Section 3.4.1 and Section 3.4.2). Two images were tested: Lena (512×512) and Woman (2048×2560; the largest in our test set). The tests for LPTCM estimation, HDQ optimization and compressing the outlier map were repeated 1000 times and the average time taken to complete each process was recorded. For SDQ optimization, the tests were repeated 10 times and the average time was recorded.

The average computation times are summarized in Table 4.3. Test results corresponding to the proposed image coder are in bold; test results corresponding to benchmarks for comparison are in regular type. Even for the largest image in our test set, LPTCM parameter estimation only takes 110 ms on average; subsequent HDQ optimization under only takes an additional 14 ms on average. These times are very promising and demonstrate that CASQ can easily be made into a practical system.

Table 4.3: Computational performance summary

| Image | Lena | Woman |
|---|---|---|
| # of blocks in image | 4096 | 81,920 |
| LPTCM estimation (ms) | **39** | **108** |
| Compressing and decompressing $m(i,j)$ | **0.5** | **8.8** |
| HDQ opt. from [7] (ms) | 1.2 | 13.5 |
| HDQ opt. proposed, $n_C = 2$ (ms) | **1.5** | **13.7** |
| SDQ opt. from [6] (Huffman) (ms) | 175 | 3900 |
| SDQ opt. proposed (Huffman), $n_C = 2$ (ms) | **208** | **3500** |
| SDQ opt. from [8] (ARL) (ms) | 220 | 3650 |
| SDQ opt. proposed (ARL), $n_C = 2$ (ms) | **201** | **3420** |

## 4.6 Summary

In this chapter, we proposed a new, practical image coder based on the CASQ framework developed in Chapter 3. We discussed an overview of the coder and then discussed each component of the coder individually. We proposed an efficient, effective online classifier for the purpose of quantization context selection. We applied the quantizer optimization solutions from Chapter 3. We then proposed some modifications to JPEG's Huffman coder and the ARL coder from [1] for compatibility with CASQ, thus completing the description of our image coder. Finally, we discussed some of the experiments that we ran to confirm both the superior RD performance and practical computational performance of our image coder.

# Chapter 5

# Conclusion and Future Work

## 5.1 Conclusion

In this thesis, we discussed the concept of optimal quantizer design in JPEG. We pointed out a serious shortcoming of JPEG, in that it lacks local adaptivity even when optimal quantizer design is achieved. We discussed several previous attempts at optimal quantizer design in both the HDQ and SDQ settings for JPEG, including current state-of-the-art methods. Motivated by this shortcoming of JPEG, we proposed a new kind of image quantization framework called context adaptive space quantization (CASQ), which attempts to address this shortcoming by quantizing blocks of an image conditioned on a quantization context. We subsequently formulated and solved the problem of optimal quantizer design in both the HDQ and SDQ settings under CASQ (based heavily on OptD theory from [7]), in addition to providing efficient algorithms for our solutions based on previous works. Finally, we proposed a practical image coder based on CASQ, which was shown to achieve superior RD performance compared to the current state-of-the art methods with very little additional computational complexity.

## 5.2 Applications and Future Work

### 5.2.1 Video Coding

While Chapter 4 was entirely dedicated to image coding, it was based on the framework developed in Chapter 3 which does not, in fact, have any inherent limitations to still image-

only coding. It is believed that CASQ could be applied quite readily to video coding, especially intra-frame coding (which is essentially still image coding). It could also be applied to the residue images of inter-frame coding, where – depending on the goodness of frame prediction, motion estimation, etc. – there are typically very large regions of little to no detail. The main issues with applying CASQ to video coding are related to the syntax of the quantizers, which makes it very non-obvious how the sources $\{X_i\}$ (Section 3.3) should be chosen. Recent video coders, such as HEVC, allow some flexibility in quantizer design that can be taken advantage of in CASQ. However, the overhead of defining custom quantizers can be significant depending on how the random sources $\{X_i\}$ are chosen. The potential applications of CASQ to video coding are currently being investigated.

### 5.2.2 Optimal Context Selection

Context selection is crucial to the performance of CASQ. There are two free parameters: the number of contexts $n_C$ and the selection of contexts for each block. Due to the overhead incurred from having many contexts defined, the best choice of $n_C$ is most likely a small number. Thus, the more important free parameter here is context selection itself.

Suppose that context selection was performed at random, where two contexts are defined. Then there is no reason to believe that any two blocks associated with the same context will bear any resemblance to each other in any way. But the idea behind CASQ is to group together blocks which have similar amounts of detail, because blocks with little detail benefit more in the RD sense from quantizers with coarse step sizes; on the other hand, blocks with lots of detail benefit more in the RD sense from quantizers with fine step sizes. Therefore, randomized context selection will most likely perform poorly.

This was the motivation behind our LPTCM-based classifier from Chapter 4: it incorporates image understanding, in some sense, for the purpose of grouping blocks which contribute significantly to the perceptual details of the image. It turns out that this works rather well in our image coder, yielding significant gains.

However, our classifier is also greedy and has no feedback; that is, it has no idea if the classification it produces is actually going to perform well on an image-by-image basis. For this reason, we consider optimal context selection as a potential future work.

One way to approach this problem would be to add a simple feedback loop to the system. Since CASQ optimization runs quite fast in practice, then a few dozen iterations of a feedback loop could be performed in practice as well. This feedback loop would involve some kind of "goodness" measure of the classifier based on the resulting rate and distortion after quantization (and possibly entropy coding for higher accuracy, since it runs

quite fast as well). Then, based on this goodness measure, an update could be applied to the classifier in hopes that it will incrementally improve the RD performance upon a subsequent iteration of running the image coder.

Another way to approach this problem is to formulate it as a constrained joint optimization problem over all quantizers and classifiers:

$$\min_{L,Q,Q'_Q} R(L,Q,Q'_Q) \quad \text{s.t.} \quad D(L,Q,Q'_Q) \leq D_T \tag{5.1}$$

where the classifier is denoted by $L$, the quantization table is denoted by $Q$, the mapping between transform coefficient sequences and quantized index sequences is denoted by $Q'_Q$, the bit rate resulting from compressing the image using $L$, $Q$ and $Q'_Q$ is denoted by $R(L,Q,Q'_Q)$, the corresponding distortion is denoted by $D(L,Q,Q'_Q)$, and the distortion budget is denoted by $D_T$. A brute-force solution would have exponential complexity with the size of the image and would not be tractable. It would be promising to investigate solutions to this problem as a future work.

#### Human Visual System

Ultimately, there is no one "correct" way to perform context selection. In the above discussion, we defined the "best" context selection as the one which performs the best in the RD sense. An alternative approach to optimizing context selection would be to incorporate the human visual system (HVS), where the "best" context selection is instead the one which results in an image that is the most pleasing to the human eye. One way to do this is to use an HVS-based distortion measure for $D(\cdot)$ in the minimization problem from earlier; for example, the structural similarity (SSIM) index measure [16]. Another method that may be applicable to context selection is the use of visual saliency modelling, which has the ability to identify parts of an image that draw the most attention; it may be argued that such parts of the image should be placed into the non-homogeneous region for CASQ, for example. It would be interesting to investigate the incorporation of the HVS as a future work, considering that the HVS plays a very important role in how we perceive imagery in all forms.

### 5.2.3 Wavelet Compression

JPEG is a block-based frequency transform coder. An alternative method to this kind of compression is wavelet compression, where an image is recursively decomposed into so-called "sub-bands" using the discrete wavelet transform (DWT). Typically, an image is

filtered in both the horizontal and vertical directions using both low-pass and high-pass filters, resulting in four sub-bands; the sub-band corresponding to the low-pass filter in both directions may be recursively decomposed to an arbitrary number of levels. This is the type of compression used in JPEG2000.

CASQ may be applied to wavelet compression, as long as uniform quantization is used and as long as there exists a mechanism for quantizing different regions of an image using different step sizes. The performance of CASQ when applied to wavelet compression, however, depends on how well the distributions of coefficients associated with the random sources $\{X_i\}$ in CASQ can be modelled. For sub-bands where a high-pass filter is used, the coefficients tend to be distributed similar to the case of AC DCT coefficients; only the coefficients from low-pass sub-bands need special consideration. If more DWT levels are applied, then exponentially fewer coefficients will belong to the low-pass sub-band, potentially allowing for even better modelling overall compared to DCT coding – where, in the case of JPEG, DC coefficients always comprise 1/64th of the total coefficients. Therefore, from a distribution modelling standpoint, we expect CASQ to apply reasonably well to wavelet compression.

In JPEG2000, however, only one quantization step size may be specified for each sub-band, and all coefficients within that sub-band must be quantized using that step size; hence, it is not really possible to provide the local separation of coefficients into random sources $\{X_i\}$ that CASQ requires. In fact, the RD performance trade-off in JPEG2000 comes primarily from entropy coding rather than from quantization. Hence, CASQ would probably not apply very well to JPEG2000 due to its coding syntax.

### 5.2.4 Entropy Coding

In Chapter 4, we proposed a practical image coding system based on CASQ which involved simple modifications to standard JPEG Huffman coding and ARL coding. We know from [1] that ARL outperforms Huffman coding by a significant margin. We also observed that the gain margin achieved by our coder was higher with ARL coding than with Huffman coding; thus, it may seem that having a more efficient entropy coder than ARL may improve our gain margins even further. Recall from (3.20) that the accuracy of OptD from [7] (and, by extension, CASQ) depends on the efficiency of the entropy coder. Therefore, it is reasonably believed that investigation into better entropy coders would be valuable as a future work.

# Appendix A

# Detailed Experimental Results

## A.1 RD Tables

Table A.1: PSNR Performance using Huffman Coding for Lena

| HDQ-JPEG | | HDQ-OptD | | HDQ-CASQ (proposed) | | SDQ | | SDQ-CASQ (proposed) | | JPEG2000 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* |
| 0.250 | 31.61 | 0.249 | 31.84 | 0.250 | **32.23** | 0.251 | 32.34 | 0.251 | **32.45** | 0.250 | 33.22 |
| 0.498 | 34.82 | 0.500 | 35.50 | 0.500 | **35.76** | 0.501 | 35.96 | 0.501 | **36.01** | 0.500 | 36.31 |
| 0.750 | 36.59 | 0.750 | 37.72 | 0.750 | **37.86** | 0.754 | 38.08 | 0.751 | **38.09** | 0.748 | 38.11 |
| 1.044 | 37.98 | 1.000 | 39.27 | 1.000 | **39.39** | 1.000 | 39.61 | 1.000 | **39.64** | 0.999 | 39.30 |
| 1.249 | 38.90 | 1.251 | 40.52 | 1.250 | **40.66** | 1.250 | 40.89 | 1.249 | **40.93** | 1.244 | 40.35 |
| 1.504 | 39.81 | 1.504 | 41.68 | 1.501 | **41.79** | 1.500 | 42.03 | 1.500 | **42.10** | 1.487 | 41.26 |
| 1.749 | 40.66 | 1.754 | 42.76 | 1.750 | **42.89** | 1.750 | 43.15 | 1.750 | **43.24** | 1.738 | 42.35 |
| 1.968 | 41.29 | 1.995 | 43.82 | 2.000 | **44.04** | 2.000 | 44.22 | 2.000 | **44.37** | 1.995 | 43.37 |

Table A.2: PSNR Performance using Huffman Coding for Airplane

| HDQ-JPEG | | HDQ-OptD | | HDQ-CASQ (proposed) | | SDQ | | SDQ-CASQ (proposed) | | JPEG2000 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* |
| 0.250 | 30.41 | 0.246 | 30.55 | 0.250 | **31.08** | 0.250 | 31.31 | 0.251 | **31.44** | 0.249 | 31.96 |
| 0.500 | 34.32 | 0.500 | 34.95 | 0.500 | **35.42** | 0.501 | 35.51 | 0.501 | **35.73** | 0.500 | 35.99 |
| 0.753 | 36.59 | 0.750 | 37.66 | 0.750 | **38.09** | 0.749 | 38.25 | 0.751 | **38.41** | 0.747 | 38.52 |
| 0.994 | 38.25 | 1.000 | 39.84 | 1.000 | **40.18** | 1.001 | 40.32 | 1.000 | **40.43** | 0.997 | 40.25 |
| 1.269 | 39.73 | 1.250 | 41.56 | 1.250 | **41.81** | 1.250 | 41.99 | 1.250 | **42.08** | 1.249 | 41.67 |
| 1.505 | 40.79 | 1.502 | 42.98 | 1.500 | **43.27** | 1.501 | 43.45 | 1.499 | **43.55** | 1.500 | 43.04 |
| 1.735 | 41.92 | 1.747 | 44.28 | 1.749 | **44.59** | 1.751 | 44.75 | 1.750 | **44.89** | 1.746 | 44.12 |
| 2.002 | 42.86 | 2.000 | 45.48 | 2.000 | **45.82** | 1.999 | 45.96 | 2.001 | **46.15** | 1.998 | 44.91 |

Table A.3: PSNR Performance using Huffman Coding for Goldhill

| HDQ-JPEG | | HDQ-OptD | | HDQ-CASQ (proposed) | | SDQ | | SDQ-CASQ (proposed) | | JPEG2000 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* |
| 0.250 | 29.24 | 0.250 | 29.68 | 0.250 | **29.67** | 0.251 | 29.96 | 0.251 | **29.91** | 0.248 | 30.07 |
| 0.498 | 31.65 | 0.500 | 32.28 | 0.500 | **32.35** | 0.501 | 32.64 | 0.501 | **32.63** | 0.499 | 32.70 |
| 0.750 | 33.19 | 0.750 | 34.22 | 0.750 | **34.25** | 0.751 | 34.55 | 0.750 | **34.54** | 0.747 | 34.54 |
| 1.011 | 34.51 | 0.999 | 35.84 | 1.000 | **35.86** | 1.001 | 36.12 | 1.001 | **36.11** | 0.989 | 35.87 |
| 1.249 | 35.56 | 1.250 | 37.23 | 1.250 | **37.25** | 1.251 | 37.54 | 1.251 | **37.52** | 1.250 | 37.39 |
| 1.500 | 36.54 | 1.500 | 38.53 | 1.500 | **38.53** | 1.501 | 38.84 | 1.500 | **38.84** | 1.498 | 38.46 |
| 1.762 | 37.56 | 1.752 | 39.72 | 1.750 | **39.72** | 1.750 | 40.07 | 1.751 | **40.08** | 1.749 | 39.55 |
| 2.071 | 38.54 | 2.000 | 40.90 | 1.998 | **40.93** | 2.000 | 41.29 | 2.000 | **41.28** | 1.998 | 40.67 |

Table A.4: PSNR Performance using Huffman Coding for Bike

| HDQ-JPEG | | HDQ-OptD | | HDQ-CASQ (proposed) | | SDQ | | SDQ-CASQ (proposed) | | JPEG2000 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* |
| 0.250 | 27.21 | 0.250 | 27.17 | 0.250 | **27.85** | 0.250 | 28.07 | 0.250 | **28.20** | 0.250 | 29.05 |
| 0.505 | 30.47 | 0.500 | 31.25 | 0.500 | **31.96** | 0.501 | 32.14 | 0.501 | **32.41** | 0.500 | 32.93 |
| 0.750 | 32.64 | 0.750 | 34.04 | 0.750 | **34.54** | 0.751 | 34.80 | 0.751 | **35.07** | 0.750 | 35.38 |
| 0.998 | 34.37 | 1.000 | 36.19 | 1.000 | **36.69** | 1.001 | 36.89 | 1.001 | **37.16** | 1.000 | 37.32 |
| 1.303 | 36.07 | 1.250 | 38.07 | 1.250 | **38.46** | 1.251 | 38.68 | 1.253 | **38.89** | 1.250 | 38.96 |
| 1.500 | 37.25 | 1.499 | 39.68 | 1.500 | **40.09** | 1.500 | 40.26 | 1.500 | **40.42** | 1.500 | 40.27 |
| 1.750 | 38.43 | 1.749 | 41.21 | 1.750 | **41.47** | 1.750 | 41.63 | 1.750 | **41.84** | 1.750 | 41.45 |
| 2.000 | 39.61 | 2.000 | 42.47 | 2.000 | **42.79** | 1.999 | 42.93 | 2.000 | **43.16** | 2.000 | 42.77 |

Table A.5: PSNR Performance using Huffman Coding for Woman

| HDQ-JPEG | | HDQ-OptD | | HDQ-CASQ (proposed) | | SDQ | | SDQ-CASQ (proposed) | | JPEG2000 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* |
| 0.250 | 27.99 | 0.250 | 28.28 | 0.250 | **28.93** | 0.251 | 29.08 | 0.252 | **29.30** | 0.250 | 29.27 |
| 0.500 | 30.89 | 0.500 | 32.02 | 0.500 | **32.61** | 0.504 | 32.76 | 0.505 | **33.01** | 0.500 | 32.85 |
| 0.773 | 33.07 | 0.750 | 34.75 | 0.750 | **35.23** | 0.753 | 35.41 | 0.755 | **35.66** | 0.750 | 35.45 |
| 1.003 | 34.62 | 1.000 | 36.90 | 1.000 | **37.38** | 1.005 | 37.58 | 1.004 | **37.73** | 1.000 | 37.52 |
| 1.273 | 36.12 | 1.250 | 38.62 | 1.250 | **39.09** | 1.254 | 39.28 | 1.252 | **39.42** | 1.250 | 39.15 |
| 1.509 | 37.48 | 1.498 | 40.02 | 1.500 | **40.55** | 1.501 | 40.72 | 1.502 | **40.90** | 1.500 | 40.46 |
| 1.745 | 38.68 | 1.749 | 41.34 | 1.750 | **41.91** | 1.754 | 42.02 | 1.755 | **42.22** | 1.750 | 41.55 |
| 2.000 | 39.88 | 2.000 | 42.58 | 1.999 | **43.15** | 2.001 | 43.19 | 2.003 | **43.43** | 2.000 | 42.76 |

Table A.6: PSNR Performance using Huffman Coding for Stockholm

| HDQ-JPEG | | HDQ-OptD | | HDQ-CASQ (proposed) | | SDQ | | SDQ-CASQ (proposed) | | JPEG2000 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* |
| 0.250 | 29.32 | 0.250 | 29.65 | 0.250 | **29.72** | 0.249 | 30.00 | 0.251 | **30.05** | 0.249 | 30.15 |
| 0.500 | 32.00 | 0.501 | 32.69 | 0.500 | **32.73** | 0.505 | 33.00 | 0.504 | **33.01** | 0.499 | 32.94 |
| 0.751 | 33.63 | 0.750 | 34.62 | 0.750 | **34.65** | 0.755 | 34.87 | 0.755 | **34.88** | 0.750 | 34.72 |
| 0.993 | 34.81 | 1.000 | 36.05 | 1.000 | **36.10** | 1.004 | 36.33 | 1.005 | **36.34** | 1.000 | 36.02 |
| 1.303 | 36.00 | 1.251 | 37.30 | 1.250 | **37.34** | 1.253 | 37.60 | 1.253 | **37.62** | 1.250 | 37.35 |
| 1.500 | 36.75 | 1.500 | 38.49 | 1.500 | **38.55** | 1.504 | 38.78 | 1.505 | **38.82** | 1.500 | 38.27 |
| 1.749 | 37.61 | 1.751 | 39.64 | 1.750 | **39.70** | 1.752 | 39.93 | 1.754 | **39.97** | 1.746 | 39.30 |
| 2.041 | 38.41 | 2.001 | 40.80 | 1.999 | **40.87** | 2.001 | 41.08 | 2.005 | **41.13** | 2.000 | 40.49 |

Table A.7: PSNR Performance using Huffman Coding for Kimono

| HDQ-JPEG | | HDQ-OptD | | HDQ-CASQ (proposed) | | SDQ | | SDQ-CASQ (proposed) | | JPEG2000 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* |
| 0.250 | 39.71 | 0.250 | 39.98 | 0.250 | **40.12** | 0.254 | 40.35 | 0.255 | **40.36** | 0.250 | 41.21 |
| 0.504 | 42.80 | 0.498 | 43.05 | 0.496 | **43.04** | 0.503 | 43.20 | 0.505 | **43.19** | 0.499 | 42.91 |
| 0.746 | 43.94 | 0.745 | 44.29 | 0.753 | **44.31** | 0.755 | 44.48 | 0.752 | **44.45** | 0.750 | 43.78 |
| 1.032 | 44.76 | 1.001 | 45.33 | 0.997 | **45.30** | 1.002 | 45.54 | 1.004 | **45.54** | 1.000 | 44.43 |
| 1.228 | 45.24 | 1.252 | 46.30 | 1.245 | **46.26** | 1.247 | 46.56 | 1.250 | **46.57** | 1.248 | 45.08 |
| 1.456 | 45.71 | 1.507 | 47.39 | 1.499 | **47.34** | 1.502 | 47.64 | 1.502 | **47.63** | 1.498 | 45.77 |
| 1.747 | 46.29 | 1.758 | 48.41 | 1.740 | **48.36** | 1.755 | 48.61 | 1.751 | **48.60** | 1.749 | 47.28 |
| 1.994 | 47.03 | 2.010 | 49.31 | 2.003 | **49.26** | 1.999 | 49.72 | 2.004 | **49.73** | 1.998 | 48.08 |

Table A.8: PSNR Performance using ARL Coding for Lena

| HDQ-JPEG | | HDQ-OptD | | HDQ-CASQ (proposed) | | SDQ | | SDQ-CASQ (proposed) | | JPEG2000 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* |
| 0.250 | 32.19 | 0.250 | 32.52 | 0.250 | **32.89** | 0.250 | 32.88 | 0.250 | **33.05** | 0.250 | 33.22 |
| 0.497 | 35.27 | 0.500 | 36.02 | 0.500 | **36.36** | 0.500 | 36.39 | 0.500 | **36.52** | 0.500 | 36.31 |
| 0.752 | 36.96 | 0.750 | 38.17 | 0.750 | **38.42** | 0.750 | 38.56 | 0.750 | **38.65** | 0.748 | 38.11 |
| 1.011 | 38.23 | 1.000 | 39.70 | 1.000 | **39.93** | 1.000 | 40.10 | 1.000 | **40.17** | 0.999 | 39.30 |
| 1.254 | 39.22 | 1.250 | 40.97 | 1.250 | **41.24** | 1.250 | 41.37 | 1.250 | **41.48** | 1.244 | 40.35 |
| 1.555 | 40.22 | 1.495 | 42.09 | 1.501 | **42.44** | 1.500 | 42.49 | 1.500 | **42.68** | 1.487 | 41.26 |
| 1.748 | 40.92 | 1.746 | 43.22 | 1.749 | **43.56** | 1.750 | 43.66 | 1.750 | **43.84** | 1.738 | 42.35 |
| 1.996 | 41.65 | 1.998 | 44.43 | 1.996 | **44.75** | 2.000 | 44.82 | 2.000 | **45.14** | 1.995 | 43.37 |

Table A.9: PSNR Performance using ARL Coding for Airplane

| HDQ-JPEG | | HDQ-OptD | | HDQ-CASQ (proposed) | | SDQ | | SDQ-CASQ (proposed) | | JPEG2000 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* |
| 0.250 | 31.00 | 0.250 | 31.40 | 0.250 | **31.90** | 0.250 | 31.84 | 0.250 | **32.03** | 0.249 | 31.96 |
| 0.499 | 34.85 | 0.500 | 35.59 | 0.500 | **36.09** | 0.500 | 36.02 | 0.500 | **36.27** | 0.500 | 35.99 |
| 0.749 | 36.98 | 0.751 | 38.41 | 0.750 | **38.78** | 0.750 | 38.76 | 0.750 | **39.00** | 0.747 | 38.52 |
| 1.028 | 38.68 | 1.003 | 40.36 | 1.000 | **40.77** | 1.000 | 40.79 | 1.000 | **41.00** | 0.997 | 40.25 |
| 1.255 | 39.90 | 1.249 | 42.00 | 1.251 | **42.41** | 1.250 | 42.44 | 1.250 | **42.62** | 1.249 | 41.67 |
| 1.506 | 41.11 | 1.499 | 43.43 | 1.499 | **43.83** | 1.500 | 43.87 | 1.500 | **44.09** | 1.500 | 43.04 |
| 1.754 | 42.11 | 1.751 | 44.70 | 1.751 | **45.13** | 1.750 | 45.13 | 1.750 | **45.47** | 1.746 | 44.12 |
| 1.971 | 43.01 | 2.001 | 45.97 | 2.000 | **46.48** | 2.000 | 46.36 | 2.000 | **46.75** | 1.998 | 44.91 |

Table A.10: PSNR Performance using ARL Coding for Goldhill

| HDQ-JPEG | | HDQ-OptD | | HDQ-CASQ (proposed) | | SDQ | | SDQ-CASQ (proposed) | | JPEG2000 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* |
| 0.250 | 29.48 | 0.250 | 29.94 | 0.250 | **30.05** | 0.250 | 30.16 | 0.250 | **30.17** | 0.248 | 30.07 |
| 0.500 | 32.02 | 0.500 | 32.67 | 0.500 | **32.71** | 0.500 | 32.89 | 0.500 | **32.89** | 0.499 | 32.70 |
| 0.754 | 33.63 | 0.750 | 34.68 | 0.750 | **34.72** | 0.750 | 34.91 | 0.750 | **34.90** | 0.747 | 34.54 |
| 0.992 | 34.96 | 1.000 | 36.34 | 1.000 | **36.39** | 1.000 | 36.61 | 1.000 | **36.60** | 0.989 | 35.87 |
| 1.251 | 36.08 | 1.250 | 37.84 | 1.250 | **37.87** | 1.250 | 38.12 | 1.250 | **38.12** | 1.250 | 37.39 |
| 1.505 | 37.05 | 1.500 | 39.20 | 1.500 | **39.24** | 1.500 | 39.54 | 1.500 | **39.55** | 1.498 | 38.46 |
| 1.758 | 37.93 | 1.751 | 40.50 | 1.748 | **40.53** | 1.750 | 40.88 | 1.750 | **40.90** | 1.749 | 39.55 |
| 1.983 | 38.81 | 2.001 | 41.75 | 2.000 | **41.80** | 2.002 | 42.20 | 2.000 | **42.21** | 1.998 | 40.67 |

Table A.11: PSNR Performance using ARL Coding for Bike

| HDQ-JPEG | | HDQ-OptD | | HDQ-CASQ (proposed) | | SDQ | | SDQ-CASQ (proposed) | | JPEG2000 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* |
| 0.250 | 27.64 | 0.250 | 27.66 | 0.250 | **28.47** | 0.250 | 28.67 | 0.250 | **28.70** | 0.250 | 29.05 |
| 0.501 | 30.85 | 0.500 | 31.78 | 0.500 | **32.47** | 0.500 | 32.53 | 0.500 | **32.84** | 0.500 | 32.93 |
| 0.780 | 33.14 | 0.750 | 34.53 | 0.750 | **35.09** | 0.750 | 35.06 | 0.750 | **35.47** | 0.750 | 35.38 |
| 1.000 | 34.64 | 1.000 | 36.67 | 1.000 | **37.18** | 1.000 | 37.25 | 1.000 | **37.52** | 1.000 | 37.32 |
| 1.262 | 36.07 | 1.249 | 38.53 | 1.250 | **38.95** | 1.250 | 39.06 | 1.250 | **39.34** | 1.250 | 38.96 |
| 1.498 | 37.42 | 1.500 | 40.14 | 1.501 | **40.57** | 1.500 | 40.66 | 1.500 | **40.95** | 1.500 | 40.27 |
| 1.750 | 38.65 | 1.749 | 41.55 | 1.750 | **42.03** | 1.750 | 42.05 | 1.750 | **42.44** | 1.750 | 41.45 |
| 1.992 | 39.77 | 1.999 | 42.78 | 1.999 | **43.36** | 2.000 | 43.32 | 2.000 | **43.83** | 2.000 | 42.77 |

Table A.12: PSNR Performance using ARL Coding for Woman

| HDQ-JPEG | | HDQ-OptD | | HDQ-CASQ (proposed) | | SDQ | | SDQ-CASQ (proposed) | | JPEG2000 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* |
| 0.250 | 28.45 | 0.250 | 29.06 | 0.250 | **29.51** | 0.250 | 29.67 | 0.250 | **29.75** | 0.250 | 29.27 |
| 0.504 | 31.45 | 0.500 | 32.73 | 0.500 | **33.22** | 0.500 | 33.30 | 0.500 | **33.55** | 0.500 | 32.85 |
| 0.750 | 33.42 | 0.750 | 35.54 | 0.750 | **36.07** | 0.750 | 36.15 | 0.750 | **36.34** | 0.750 | 35.45 |
| 1.002 | 35.06 | 1.000 | 37.62 | 1.000 | **38.14** | 1.000 | 38.30 | 1.000 | **38.47** | 1.000 | 37.52 |
| 1.257 | 36.52 | 1.250 | 39.24 | 1.250 | **39.80** | 1.250 | 39.98 | 1.250 | **40.20** | 1.250 | 39.15 |
| 1.500 | 37.80 | 1.500 | 40.69 | 1.499 | **41.20** | 1.500 | 41.34 | 1.500 | **41.56** | 1.500 | 40.46 |
| 1.766 | 39.07 | 1.752 | 41.96 | 1.750 | **42.57** | 1.750 | 42.56 | 1.750 | **42.87** | 1.750 | 41.55 |
| 1.995 | 40.27 | 1.998 | 43.17 | 1.998 | **43.84** | 2.000 | 43.71 | 2.000 | **44.13** | 2.000 | 42.76 |

Table A.13: PSNR Performance using ARL Coding for Stockholm

| HDQ-JPEG | | HDQ-OptD | | HDQ-CASQ (proposed) | | SDQ | | SDQ-CASQ (proposed) | | JPEG2000 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* |
| 0.250 | 29.62 | 0.250 | 30.03 | 0.250 | **30.16** | 0.250 | 30.33 | 0.250 | **30.34** | 0.249 | 30.15 |
| 0.497 | 32.35 | 0.500 | 33.05 | 0.500 | **33.13** | 0.500 | 33.34 | 0.500 | **33.35** | 0.499 | 32.94 |
| 0.753 | 34.02 | 0.750 | 34.96 | 0.750 | **35.04** | 0.750 | 35.26 | 0.750 | **35.28** | 0.750 | 34.72 |
| 1.002 | 35.19 | 1.001 | 36.43 | 1.000 | **36.50** | 1.007 | 36.76 | 1.000 | **36.79** | 1.000 | 36.02 |
| 1.257 | 36.22 | 1.251 | 37.76 | 1.250 | **37.84** | 1.250 | 38.07 | 1.250 | **38.11** | 1.250 | 37.35 |
| 1.509 | 37.09 | 1.502 | 39.02 | 1.501 | **39.11** | 1.500 | 39.31 | 1.500 | **39.38** | 1.500 | 38.27 |
| 1.764 | 37.92 | 1.748 | 40.28 | 1.749 | **40.40** | 1.750 | 40.53 | 1.750 | **40.64** | 1.746 | 39.30 |
| 1.989 | 38.77 | 2.000 | 41.54 | 1.996 | **41.66** | 2.000 | 41.79 | 2.000 | **41.92** | 2.000 | 40.49 |

Table A.14: PSNR Performance using ARL Coding for Kimono

| HDQ-JPEG | | HDQ-OptD | | HDQ-CASQ (proposed) | | SDQ | | SDQ-CASQ (proposed) | | JPEG2000 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* | *Rate* | *PSNR* |
| 0.251 | 40.23 | 0.249 | 40.42 | 0.250 | **40.58** | 0.250 | 40.63 | 0.250 | **40.71** | 0.250 | 41.21 |
| 0.498 | 43.12 | 0.503 | 43.37 | 0.501 | **43.40** | 0.500 | 43.49 | 0.500 | **43.50** | 0.499 | 42.91 |
| 0.758 | 44.29 | 0.753 | 44.65 | 0.750 | **44.66** | 0.750 | 44.78 | 0.750 | **44.79** | 0.750 | 43.78 |
| 0.966 | 44.92 | 1.003 | 45.79 | 1.001 | **45.81** | 1.000 | 45.91 | 1.000 | **45.93** | 1.000 | 44.43 |
| 1.183 | 45.42 | 1.253 | 46.83 | 1.259 | **46.86** | 1.250 | 47.05 | 1.250 | **47.07** | 1.248 | 45.08 |
| 1.560 | 46.17 | 1.497 | 47.96 | 1.507 | **48.04** | 1.500 | 48.17 | 1.500 | **48.20** | 1.498 | 45.77 |
| 1.740 | 46.77 | 1.753 | 48.97 | 1.760 | **49.03** | 1.750 | 49.21 | 1.750 | **49.26** | 1.749 | 47.28 |
| 1.989 | 47.47 | 2.005 | 50.17 | 1.992 | **50.16** | 2.000 | 50.47 | 2.000 | **50.53** | 1.998 | 48.08 |

# References


[1] Chengjie Tu, Jie Liang, and T.D. Tran. Adaptive runlength coding. *IEEE Signal Processing Letters*, 10(3):61–64, March 2003.

[2] *Digital compression and coding of continuous-tone still images: Requirements and guidelines*, ISO/IEC 10918-1, 1993.

[3] A.C. Hung and T.H.-Y. Meng. Optimal quantizer step sizes for transform coders. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 2621–2624 vol.4, Apr 1991.

[4] Siu-Wai Wu and A. Gersho. Rate-constrained picture-adaptive quantization for jpeg baseline coders. In *International Conference on Acoustics, Speech, and Signal Processing*, volume 5, pages 389–392 vol.5, April 1993.

[5] Viresh Ratnakar and M. Livny. An efficient algorithm for optimizing dct quantization. *IEEE Transactions on Image Processing*, 9(2):267–270, Feb 2000.

[6] En-Hui Yang and Longji Wang. Joint optimization of run-length coding, huffman coding, and quantization table with complete baseline jpeg decoder compatibility. *IEEE Transactions on Image Processing*, 18(1):63–74, Jan 2009.

[7] En-Hui Yang, Chang Sun, and Jin Meng. Quantization table design revisited for image/video coding. *IEEE Transactions on Image Processing*, 23(11):4799–4811, Nov 2014.

[8] En-Hui Yang and Longji Wang. Joint optimization of run-length coding, context-based arithmetic coding and quantization step sizes. In *Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 678–681, May 2009.

[9] K. Ramchandran and M. Vetterli. Rate-distortion optimal fast thresholding with complete jpeg/mpeg decoder compatibility. *IEEE Transactions on Image Processing*, 3(5):700–704, Sep 1994.

[10] M. Crouse and K. Ramchandran. Joint thresholding and quantizer selection for transform image coding: entropy-constrained analysis and applications to baseline jpeg. *IEEE Transactions on Image Processing*, 6(2):285–297, Feb 1997.

[11] En hui Yang and Z. Zhang. The redundancy of source coding with a fidelity criterion .ii. coding at a fixed rate level with unknown statistics. *IEEE Transactions on Information Theory*, 47(1):126–145, Jan 2001.

[12] G. J. Sullivan and S. Sun. On dead-zone plus uniform threshold scalar quantization. In S. Li, F. Pereira, H.-Y. Shum, and A. G. Tescher, editors, *Visual Communications and Image Processing 2005*, volume 5960 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, pages 1041–1052, July 2005.

[13] En-Hui Yang, Xiang Yu, Jin Meng, and Chang Sun. Transparent composite model for dct coefficients: Design and analysis. *IEEE Transactions on Image Processing*, 23(3):1303–1316, March 2014.

[14] Chang Sun and En-Hui Yang. An efficient dct-based image compression system based on laplacian transparent composite model. *IEEE Transactions on Image Processing*, 24(3):886–900, March 2015.

[15] Michael Adams. The jasper project home page. `https://www.ece.uvic.ca/~frodo/jasper/`.

[16] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *Image Processing, IEEE Transactions on*, 13(4):600–612, April 2004.